An Agent Based Model for Simulating Herd Dynamics

Beth Johnson
johnsel@umd.edu

Advisor:
Dr. Bill Fagan
Department of Biology
bfagan@umd.edu

Abstract

Biology and ecology currently lack a unifying theory for movement mechanisms as well as population level patterns. Most methods for analyzing both movement and population patterns require assumptions about the type of movement or expected population patterns. An agent based model, where each animal is represented as an evolutionarily trained neural network, is used to generate relocation data for a variety of landscapes without making assumptions about movement mechanisms. This data are analyzed using spatial metrics to attempt to classify the population level patterns that have emerged.

1. Project Background

There are many different mechanisms that animals may use to direct movement, for example an animal could use sensory cues (sight, smell) to move towards a resource such as food or water. Movement mechanisms fall into three broad categories: non-oriented, oriented, and spatial memory. Animals may use multiple mechanisms, for example, a random walk around a central place for part of the year followed by a migration to a new location that is guided by memory of landmarks. While there are various models for each of the different types of mechanisms, there is not a model that can accommodate all three. Based on the spatiotemporal variability of the landscape, there are three population movement patterns that animals may exhibit: sedentary ranges, migration, and nomadism. Sedentary range is when the population moves about a central place (relatively small compared to the available range for that species), migration is when a population follows a predictable cycle of movement between two disjoint locations, and nomadism is when the population moves to multiple locations in a non-predictable manner. It is hypothesized that the spatial and temporal variability of resource landscapes will determine the most efficient movement mechanism as well as the overall population pattern, as summarized in Figure 1. While there are different analysis methods for each of the possible population distributions, most rely on an initial assumption of the type of distribution the population is exhibiting [6].

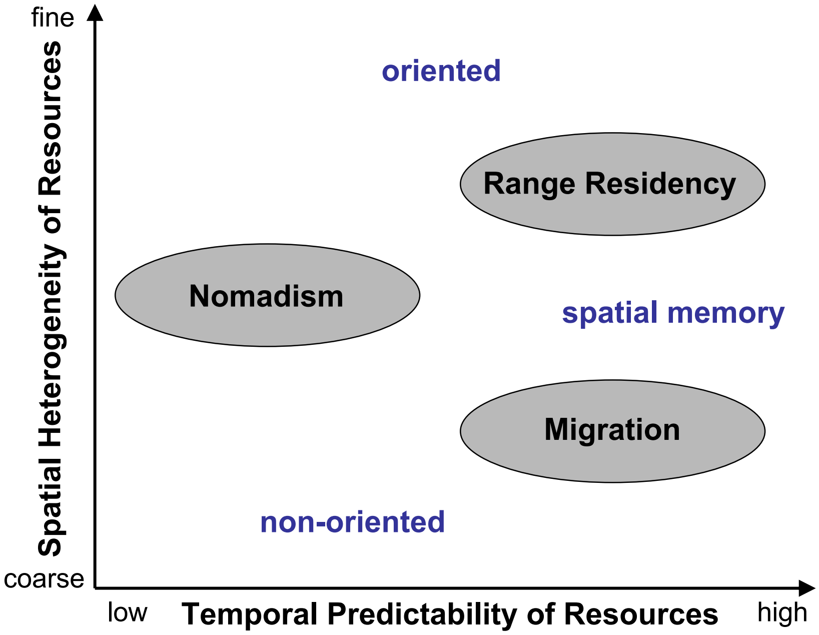


Figure 1: Hypothesized relationship between temporal predictability and spatial heterogeneity. For example, if the resources are highly predictable, spatial memory will be the most efficient mechanism. Based on the heterogeneity of the landscape, this could cause migration or range residency to occur. From [6].

2. Project Objectives

The overall objective of this paper is to be able to classify population level patterns from location data, based on three spatial metrics. To do this, an agent based model of individuals and herds will be used, where the individuals are deciding movement based on an artificial neural network and a set of movement rules.

3. Approach

There will be two parts to this project. The first part is an implementation of an agent based model with evolutionarily trained neural networks to generate relocation data. The second part is a suite of spatial metrics that will be used to analyze the simulation or relocation data to discern the type of population distribution.

Table 1: Notation used throughout.

Symbol	Meaning
N	Number of individuals (or agents)
G	Number of generations
R	Number of total resources on the grid
r	Length (in cells) of side of resource patch {1, 2, 4, 8}
Pr	Predictability of landscape {0, 25%, 50%, 75%, 100%}
P_{mut}	Probability of point mutation
P_{xover}	Probability of crossover
$\Delta x(t_i)$	Distance traveled in x direction from t_{i-1} to t_i
$\Delta y(t_i)$	Distance traveled in y direction from t_{i-1} to t_i
$P^i = \{p_1^i, p_2^i, \dots p_m^i\}$	Set of data points for individual i, where p^i is an (x, y) location
M	Total number of time steps (number of (x,y) pairs)
$S = \{s_1, s_2, \dots s_n\}$	Set of distances for which to calculate the PDI
μ_x	Global mean shift in x direction
μ_y	Global mean shift in y direction
σ	Global variance
ρ	Global correlation

3.1 Agent Based Model

The agent based model will consist of two stages – an evolution phase and a herd movement phase. The movement rules (as determined by the neural network) will be the same for both stages, with an additional rule such that agents cannot occupy the same cell at the same time step. Stage one is based on [5]. The following sections give further details about the landscape, agents, and evolution. The full algorithm of the entire agent based model can be found in Section 1 of the Appendix.

3.1.1 Landscape

As in [5], the landscape consists of a 64 by 64 grid of cells with reflective boundaries. Each cell has an integer value indicating a resource count in the cell. If an agent moves to a cell with a non-zero resource count, a resource is transferred to the agent (the agent's resource count is increased by 1, the cell's resource count is decreased by one). The landscape is varied in two ways – patch size and predictability. The patch size is defined as the size of the resource squares as they are placed on the landscape, as seen in Figure 2. Predictability is defined as the percentage of the patches that remains in the same location throughout all generations, as seen in Figure 3. The simulation will be run for all 20 possible combinations of patch size and predictability. For a given generation, the starting landscape is the same for all individuals, but the landscape changes each generation according to the predictability. For the evolution stage, the maximum resource count in any cell is 1 and if a cell is depleted it will remain depleted for the current agent and generation. For the herd movement phase, the maximum resource count in any cell will be the number of agents in the herd.

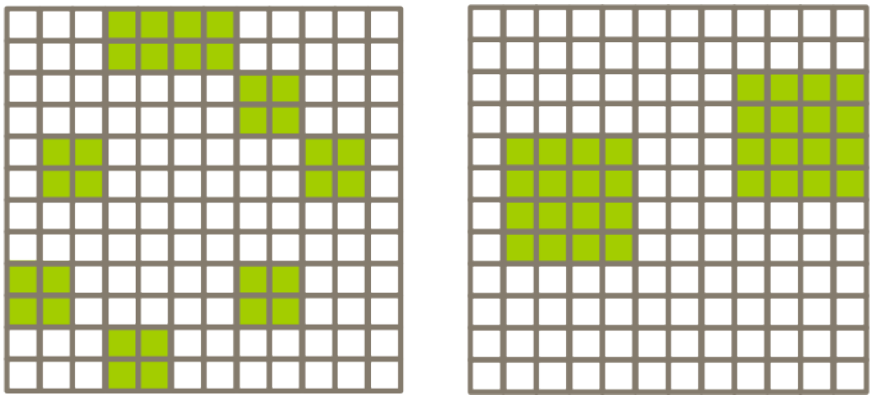


Figure 2: Patch size. The grid on the left shows a patch size of r = 2, while the right shows a patch size of r = 4. Note that the total number of resources cells remains constant, regardless of the patch size. Patch size will be {1, 2, 4, 8}.

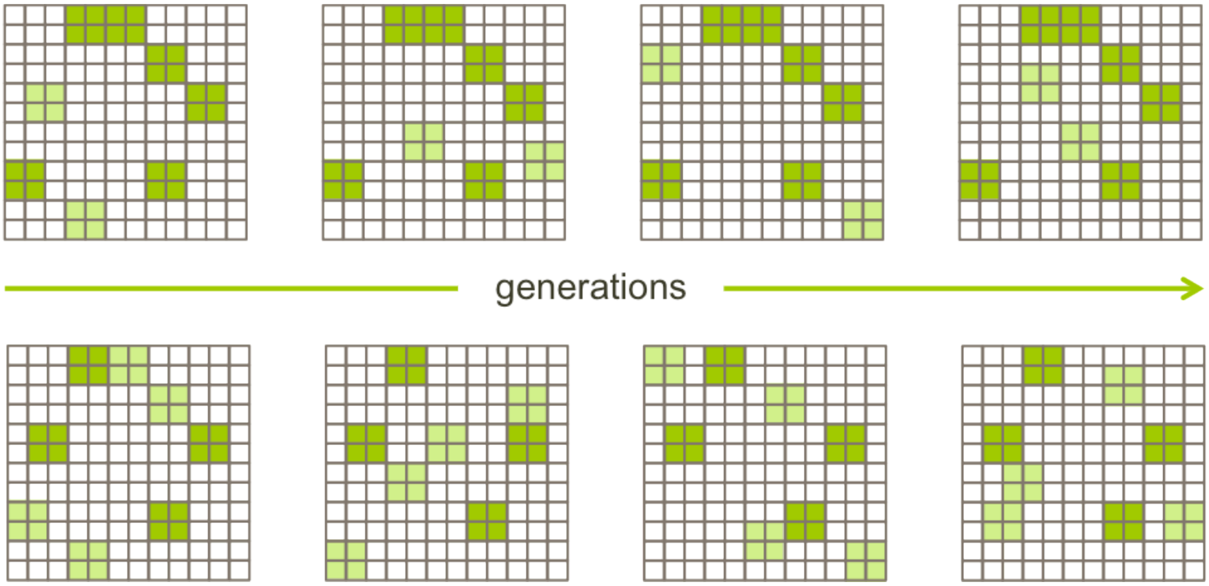


Figure 3: Landscape predictability. The top figure shows a landscape predictability of Pr = 0.75, where 6 of the 8 resource patches remain constant throughout generations. The bottom shows a landscape predictability of Pr = 0.5, where 4 of the 8 resource patches remain constant throughout generations. Landscape predictability will be selected from {0, 0.25, 0.5, 0.75, 1}.

3.1.2 Agents

Each agent is represented by a fully connected feed forward artificial neural network, consisting of seven input variables (and a bias), three hidden nodes, and three output nodes as shown in Figure 4, from [5].

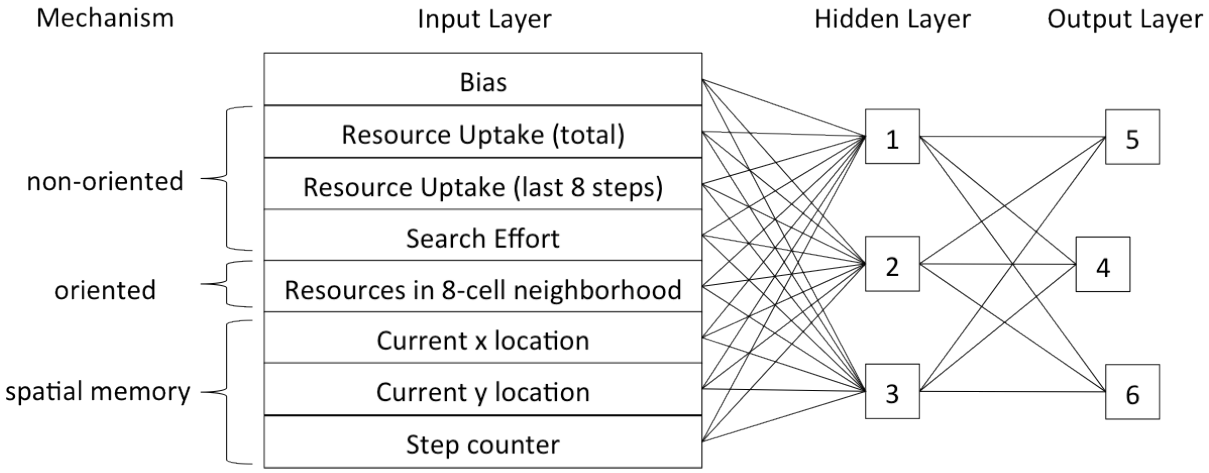


Figure 4: Agent Neural Network. There are 8 input nodes representing three different movement mechanisms. Modified from [5], with permission.

This gives 33 weights connecting the nodes, and these 33 weights are representative of the agent's "genes". The weights are integer values from [-5,5]. The initial weights for each agent are randomly selected from a uniform distribution. The inputs to the hidden layer are scaled to [-1,1], and the values of the hidden nodes and output nodes are determined by the following equations.

$$\mathrm{HiddenNode}_j = \frac{\exp\left(\sum_{i=1}^{8} \mathrm{Input_i} \times \mathrm{weight_{ij}}\right)}{1 + \exp\left(\sum_{i=1}^{8} \mathrm{Input_i} \times \mathrm{weight_{ij}}\right)}$$

$$\mathrm{OutputNode}_k = \frac{\exp\left(\sum_{j=1}^{3} \mathrm{HiddenNode_j} \times \mathrm{weight_{jk}}\right)}{1 + \exp\left(\sum_{j=1}^{3} \mathrm{HiddenNode_j} \times \mathrm{weight_{jk}}\right)}$$

Input variables are as follows and are updated after each movement step:

1. Current value of resource counter (starts at zero)
2. Increase in resource counter over the last eight steps (starts at zero)
3. Search effort over the last eight steps (defined as an agent's intention to deviate from current movement direction – see 2.1.2.3, starts at zero)
4. Number of cells in surrounding eight cell neighborhood that contains a resource
5. Current x-position
6. Current y-position
7. Current time step

Agents may move to any cell immediately adjacent and may only move one cell at any given time step. Agents may not stay in the same cell unless they are "trapped" by the edges of the landscape and other agents.

3.1.3 Evolution

Each generation, agents are randomly selected without replacement in groups of six [5]. The agent in the group of six with the largest value of the resource counter is chosen to reproduce. When the agent is reproduced, the 33 weights (representing the genes of the agent) are copied six times to create six new agents. When copying the weights, there is a 20% probability of a crossover event occurring ($P_{mut} = 0.20$) and a 2% probability at each weight for a point mutation ($P_{xover} = 0.20$) [5].

3.2 Spatial Statistics

3.2.1 Realized Mobility Index

Realized mobility index, or RMI, describes the proportion of the annual range an individual is using. The range of the individual is defined as the minimum convex polygon covering all location points for that individual. The range of the population is defined as the minimum convex polygon covering all location points for all individuals in the population. The RMI is simply the ratio of the area of the individual range to the area of the population range. An RMI is calculated for each individual in the population, and then averaged over the population [7]. The minimum convex polygon will be found using the following algorithm from Kirkpatrick and Seidel [3]. First, the upper hull will be determined, then the lower hull will be determined using the same method. If the left and right endpoints on the upper and lower hull are not the same, vertical lines can be drawn to connect them. The full algorithm can be found in Section 2 of the Appendix.

3.2.2 Population Dispersion Index

Population dispersion index, or PDI, describes how clustered the population is at different spatial scales [7]. The PDI is the average of the bivariate k-function for each individual in the population. The bivariate k-function is defined as expected number of points of pattern 1 within a distance s of an arbitrary point of pattern 2, divided by overall point density in pattern 1. The algorithm for calculating the PDI is given below. For ease of reading, the algorithm for the bivariate k-function is separated and detailed following the PDI algorithm. The algorithm for computing the bivariate k-function is based on Cressie [2] with edge corrections from Lotwick and Silverman [4]. The full algorithm can be found in Section 3 of the Appendix.

3.2.3 Movement Correlation Index

The movement correlation index, or MCI, is a measure of how much the movements of the individuals are correlated with each other. A positive number indicates the individuals move together, and a negative number indicates the individuals move away from each other. A value near zero indicates that the individuals are not correlated. The MCI assumes that the movements of the individuals are described by a discrete time based Gaussian position jump process with *N* individuals and *M* time steps. A global mean shift in the x-direction (μ_x), mean shift in the y-direction (μ_y), variance (σ^2), and correlation (ρ) are assumed for all individuals. The full algorithm can be found in Section 4 of the Appendix.

4. Implementation

The above algorithms are implemented using R due to its popularity among biologists and ecologists, as well as MATLAB and C as needed for speed considerations. The code was run on a Macbook Pro with a 2.9 GHz Intel Core i7 processor with 8 GB of RAM.

5. Databases

The agent based model generated the data for input to the PDI, RMI, and MCI algorithms. Additionally, real relocation data from gazelles will be used as input to the PDI, RMI, and MCI algorithms. The relocation data will consist of a time series of x and y locations from 36 gazelles with GPS collars. The time series for an individual gazelle ranges from 50 days to 917 days, with 8111 data points in total. The time intervals between data points vary from 1 to 25 hours and will have gaps from satellite interference or battery saving programs. Some preprocessing of this data was necessary to use with the PDI, RMI, and MCI metrics.

6. Validation

6.1 Agent Based Model

The agent based model will be validated in smaller modules as well as overall.

6.1.1 Landscape Initialization

The landscape initialization was partially validated by computing the number of cells with a resource and ensuring it equals R. The landscape initialization and update was further validated by updating the landscape many times and summing the number of resources per cell over time. The cells that were selected to be stationary resource patches throughout generations should have a resource value equal to the number of updates to the landscape, the cells selected to be stationary empty patches throughout generations should have a resource value equal to zero, and the other cells should have a resource value that is less than the number of updates but uniform across all cells that were not selected to be stationary. Visualizing this with a color map of the matrix (`imagesc` function in MATLAB) next to a matrix holding only the stationary patches should show that the distribution of the patches that are not stationary is roughly uniform. The lack of pattern in the non-stationary cells confirms that the landscape is being generated appropriately. It should also be noted that due to the definition of predictability, with a patch size of 8, there are only two resource patches, which only allows three different predictability levels for resource patches. Thus, Pr = 0 and Pr = 25% both have 0 predictable resource patches, with an appropriate number of stationary empty patches, Pr = 50% and Pr = 75% have 1 predictable resource patch, with an appropriate number of stationary empty patches, and Pr = 100% has 2 predictable resource patches. The landscape validation plots can be seen in Figure 5 to Figure 9 below.

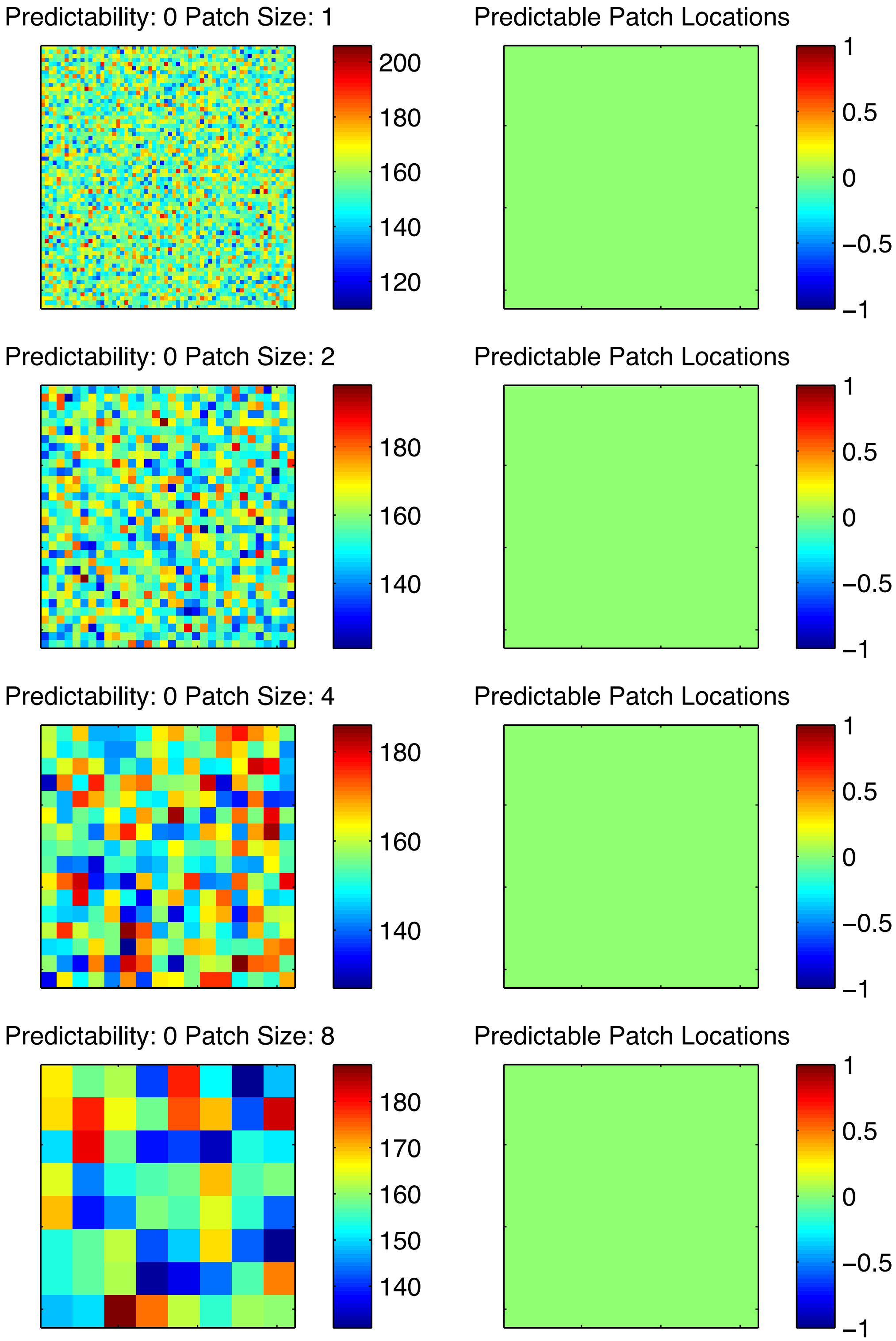


Figure 5: Landscape Generation for 5000 Runs, Pr = 0%, r = {1, 2, 4, 8}. Stable patches are shown on the right (in this case, none).

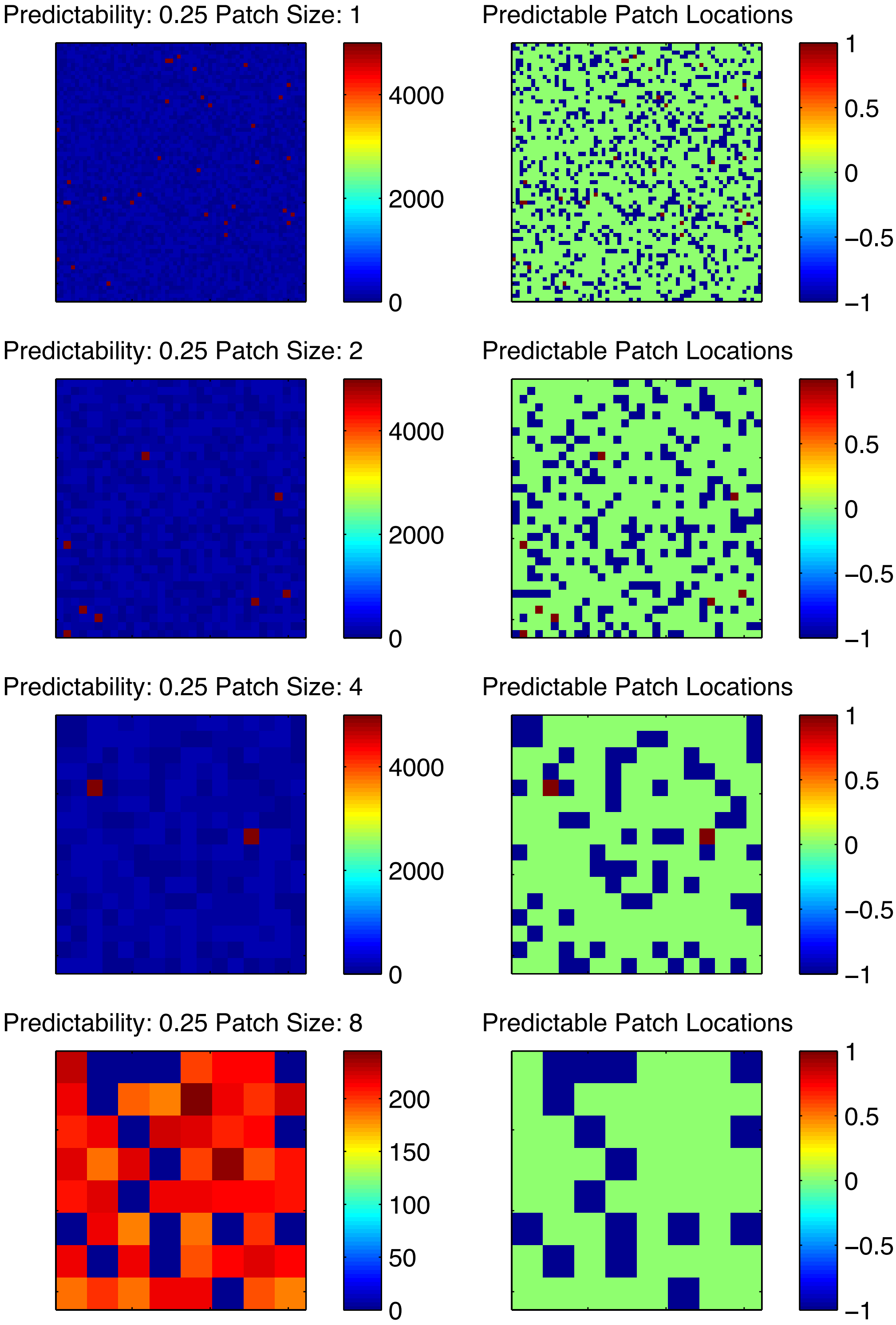


Figure 6: Landscape Generation for 5000 Runs, Pr = 25%, r = {1, 2, 4, 8}. Stable patches are shown on the right, with stable resource patches in red and stable empty patches in blue.

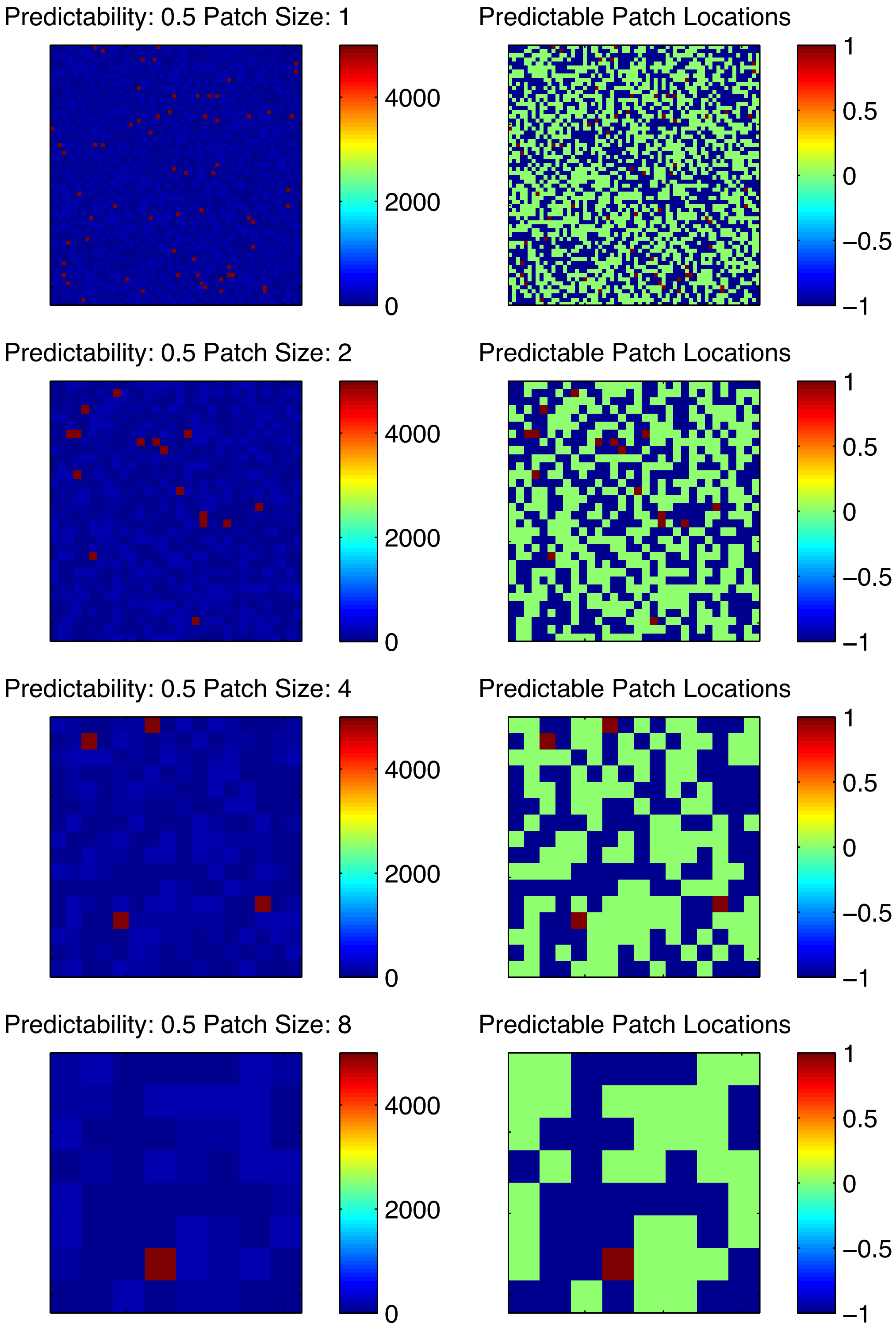


Figure 7: Landscape Generation for 5000 Runs, Pr = 50%, r = {1, 2, 4, 8}. Stable patches are shown on the right, with stable resource patches in red and stable empty patches in blue.

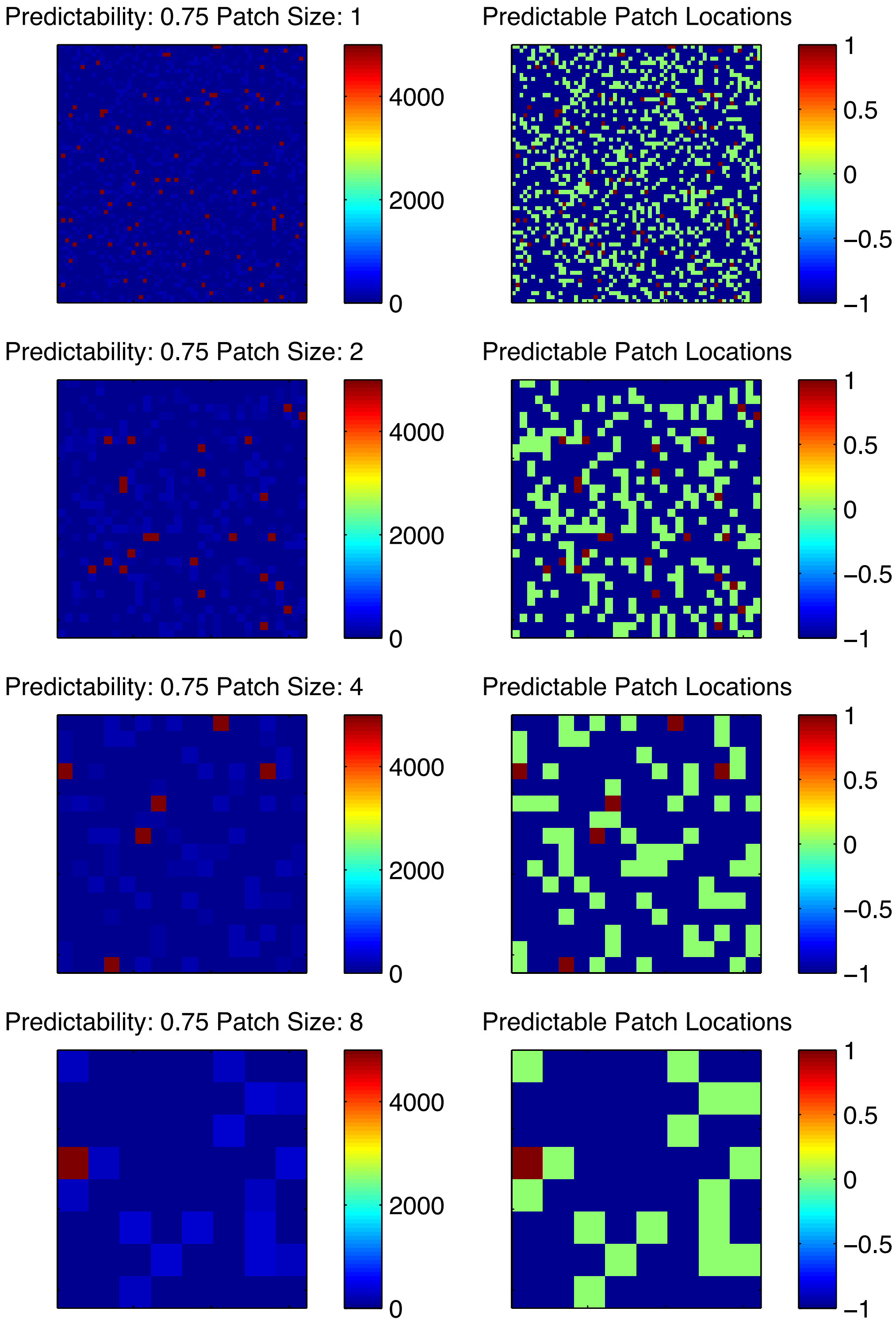


Figure 8: Landscape Generation for 5000 Runs, Pr = 75%, r = {1, 2, 4, 8}. Stable patches are shown on the right, with stable resource patches in red and stable empty patches in blue.

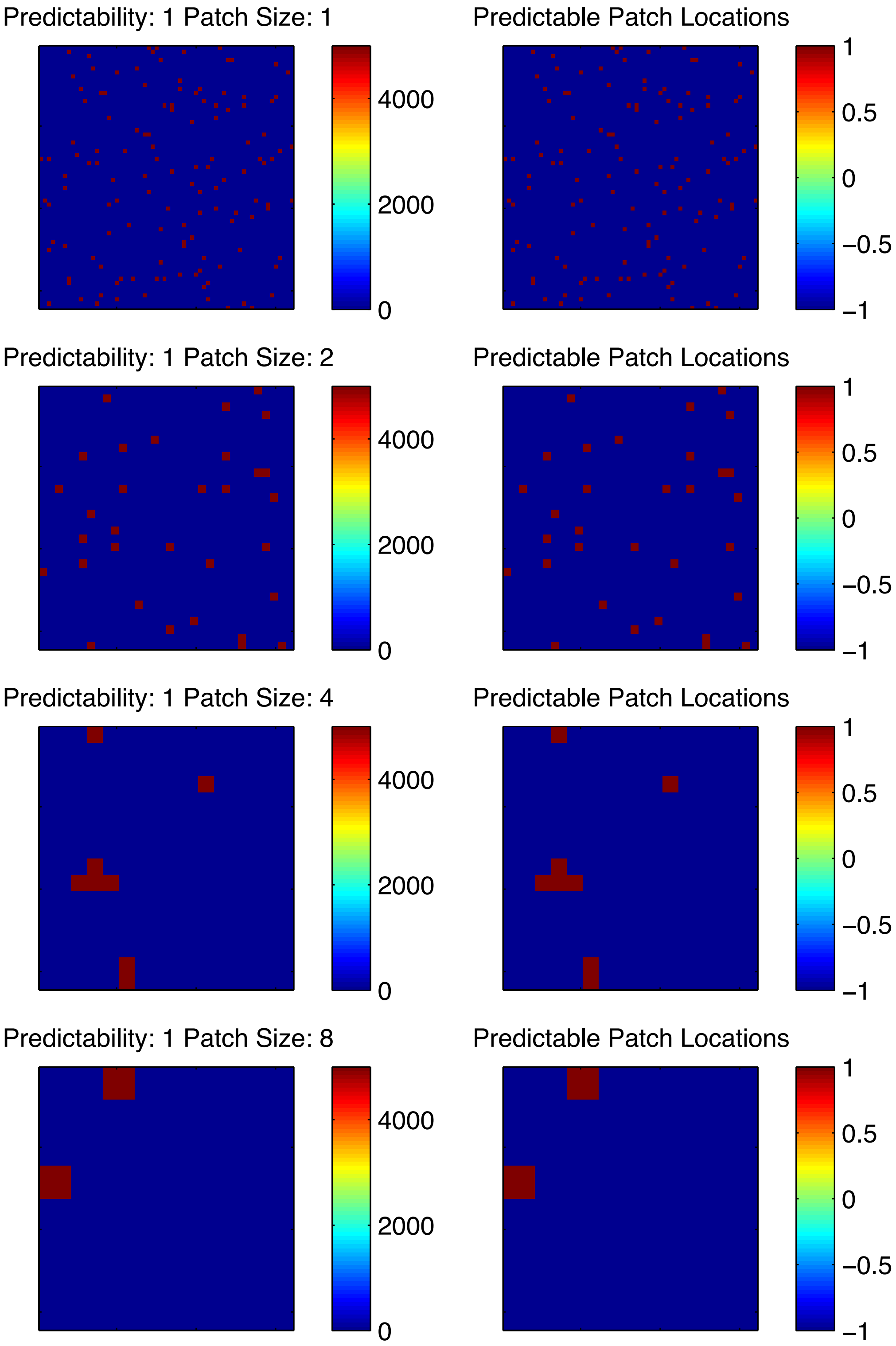


Figure 9: Landscape Generation for 5000 Runs, Pr = 100%, r = {1, 2, 4, 8}. Stable patches are shown on the right, with stable resource patches in red and stable empty patches in blue. Note that because it is 100% predictable the predictable patch locations plot exactly matches the plot from 5000 runs.

6.1.2 Agent Movement

The agent movement was validated by selecting test cases, computing values of the nodes and determining the resulting movement decisions by hand, and then comparing this with the direction selected by the implementation. For all test cases, 3 agents were initialized with specific weights, input values, and previous direction. The location and visible resources were varied throughout the test cases. Focus was given to the edges of the map in selecting locations. For each test case, the values of the hidden and output nodes were calculated by hand, and then the agent direction selection algorithm was followed. For some combinations of node values, location, and resources, the algorithm specifies multiple directions from which to randomly select. The agent direction function was called 10,000 times for each test case, and a histogram was created from the distribution of directions selected. The histograms confirm that the correct direction was always selected by the agent direction function, as well as a uniform distribution between multiple directions if multiple directions were allowed for the resulting node values. In the following figures, the location of the agent, previous direction, calculated node values, and an image of the 8-cell neighborhood are given. The x represents the agent's location, the grey shaded cells represent that the map does not exist in those cells, 0 represents no resource in that cell, and 1 represents a resource is in that cell.

Test Case 1

Row	1
Column	64
Previous Dir	2

▨	▨	▨
0	x	▨
0	0	▨

Node 4	0.810
Node 5	0.906
Node 6	0.004

Memory Movement, direction of 1, but unavailable, so randomly select from directions 5, 6, 7

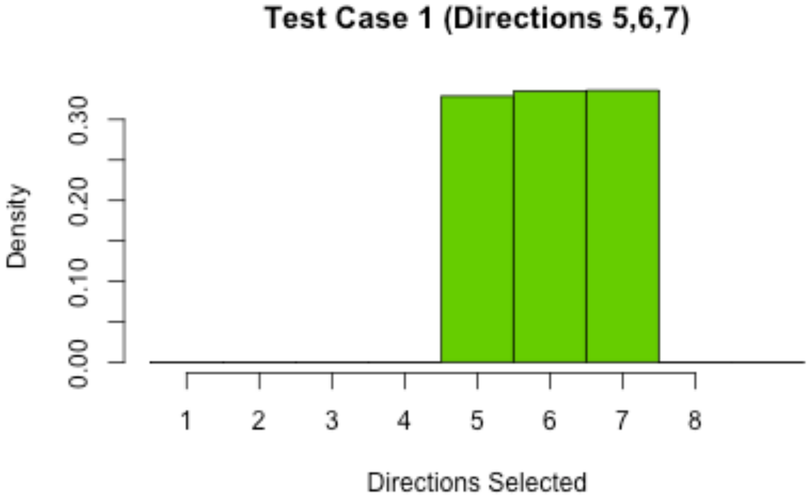


Figure 10: Agent Movement Validation Test Case 1.

Test Case 2

Row	64
Column	1
Previous Dir	6

▨	0	0
▨	x	0
▨	▨	▨

Node 4	0.981
Node 5	0.003
Node 6	0.053

Memory Movement, direction of 1

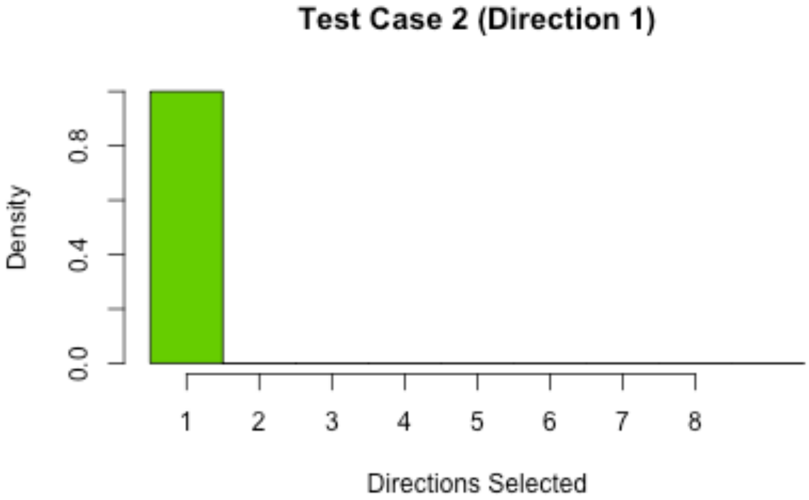


Figure 11: Agent Movement Validation Test Case 2.

Test Case 3

Row	64
Column	64
Previous Dir	4

1	1	▨
1	x	▨
▨	▨	▨

Node 4	0.501
Node 5	0.501
Node 6	0.490

Oriented Movement, 3 resources visible, select from 1, 7, 8

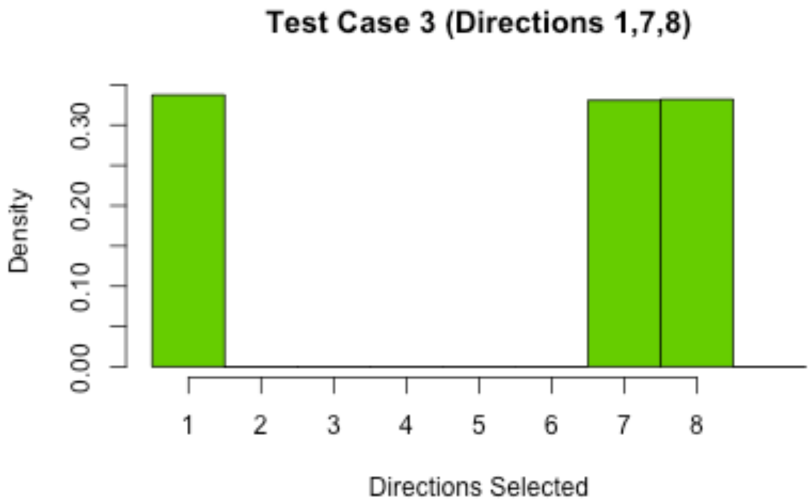

Test Case 3 (Directions 1,7,8)

Figure 12: Agent Movement Validation Test Case 3.

Test Case 4

Row	64
Column	64
Previous Dir	4

0	0	▨
1	x	▨
▨	▨	▨

Node 4	0.501
Node 5	0.501
Node 6	0.490

Oriented Movement, 1 resource visible, direction of 7

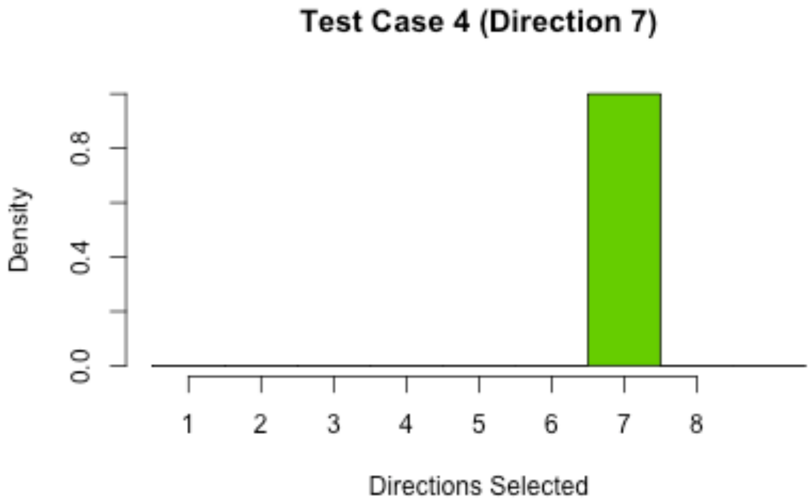

Test Case 4 (Direction 7)

Figure 13: Agent Movement Validation Test Case 4.

Test Case 5

Row	2
Column	64
Previous Dir	1

0	0	▨
0	x	▨
0	0	▨

Node 4	0.267
Node 5	0.504
Node 6	0.048

Non-Oriented Movement, previous direction was 1 - 2 is unavailable, so select from 1, 8

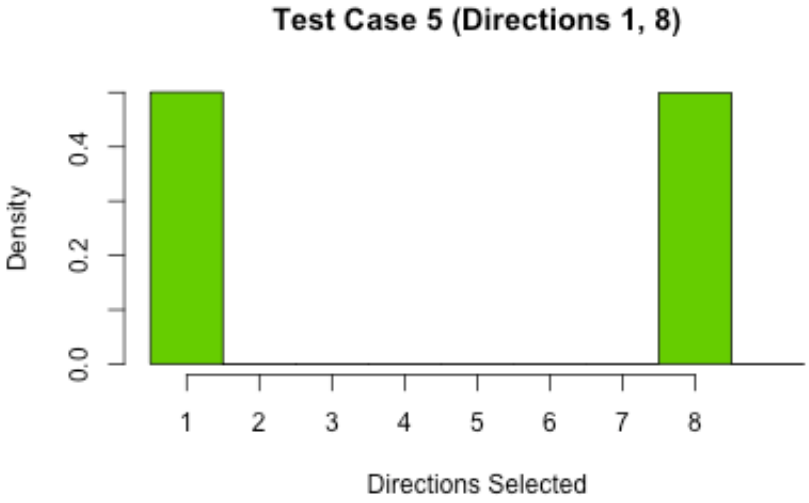


Figure 14: Agent Movement Validation Test Case 5.

Test Case 6

Row	64
Column	2
Previous Dir	6

0	0	0
0	x	0
▨	▨	▨

Node 4	0.269
Node 5	0.501
Node 6	0.047

Non-Oriented Movement, previous direction was 6 - 5 and 6 are unavailable, direction of 7

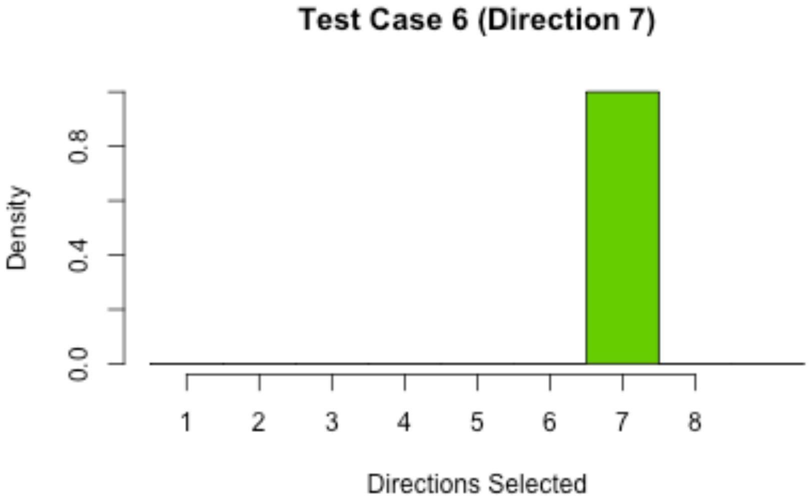


Figure 15: Agent Movement Validation Test Case 6.

Test Case 7

Row	1
Column	64
Previous Dir	2

▨	▨	▨
0	x	▨
0	0	▨

Node 4	0.267
Node 5	0.504
Node 6	0.048

Non-Oriented Movement, previous direction was 2 - 1, 2, 3, 4, 8 unavailable, so select from 5, 6, 7

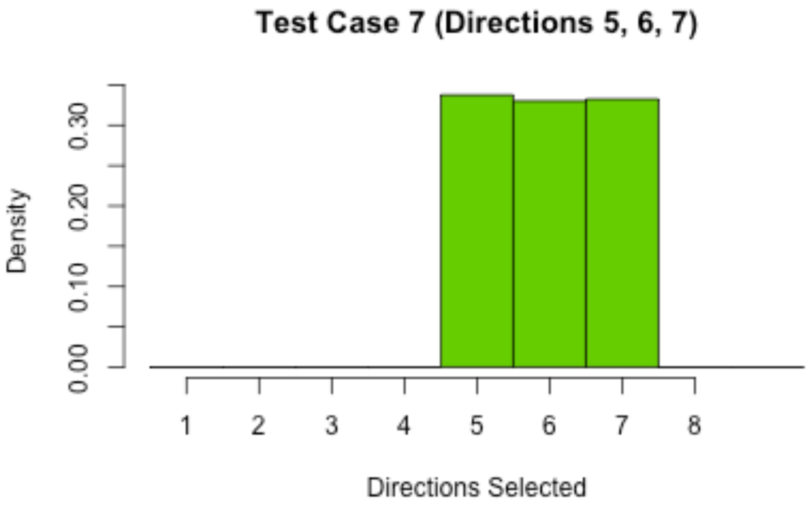

Test Case 7 (Directions 5, 6, 7)

Figure 16: Agent Movement Validation Test Case 7.

Test Case 8

Row	63
Column	63
Previous Dir	1

0	0	0
0	x	0
0	0	0

Node 4	0.267
Node 5	0.484
Node 6	0.055

Non-Oriented Movement, previous direction was 1 - select from 1, 2, 8

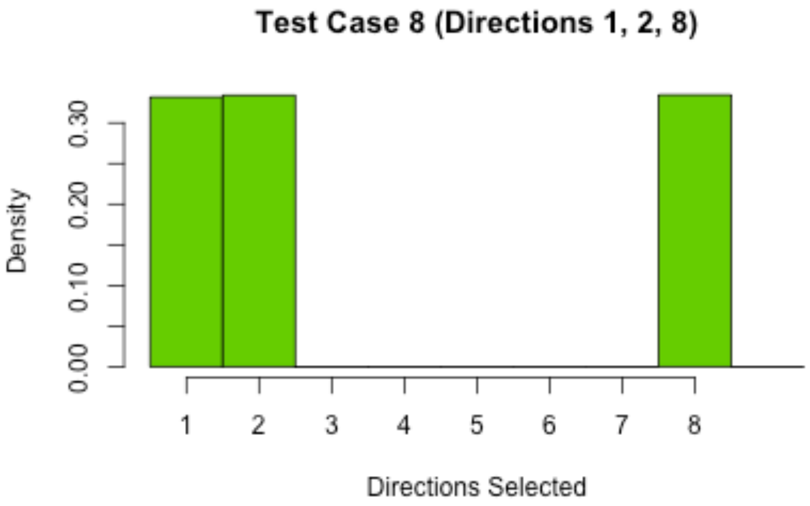

Test Case 8 (Directions 1, 2, 8)

Figure 17: Agent Movement Validation Test Case 8.

Test Case 9

Row	35
Column	35
Previous Dir	1

0	1	0
0	x	0
0	0	0

Node 4	0.500
Node 5	0.952
Node 6	0.050

Oriented Movement, 1 resource available, direction of 1

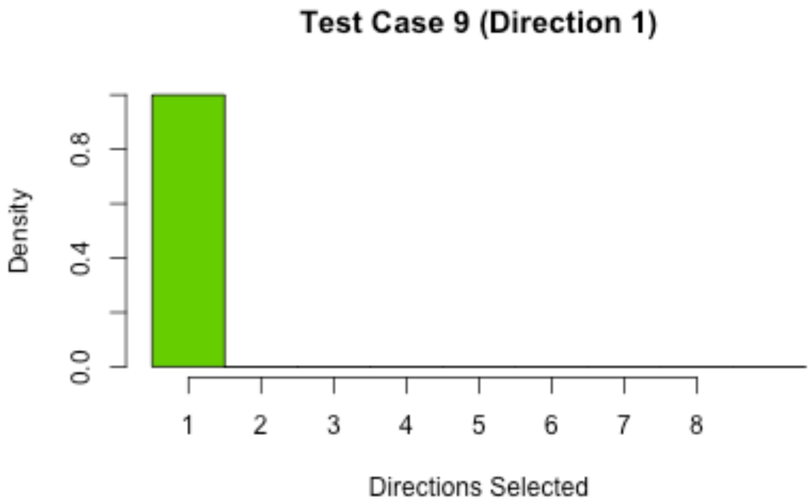


Figure 18: Agent Movement Validation Test Case 9.

Test Case 10

Row	35
Column	35
Previous Dir	1

1	1	1
0	x	0
0	0	0

Node 4	0.500
Node 5	0.952
Node 6	0.050

Oriented Movement, 3 resources available, randomly select from direction of 1, 2, 8

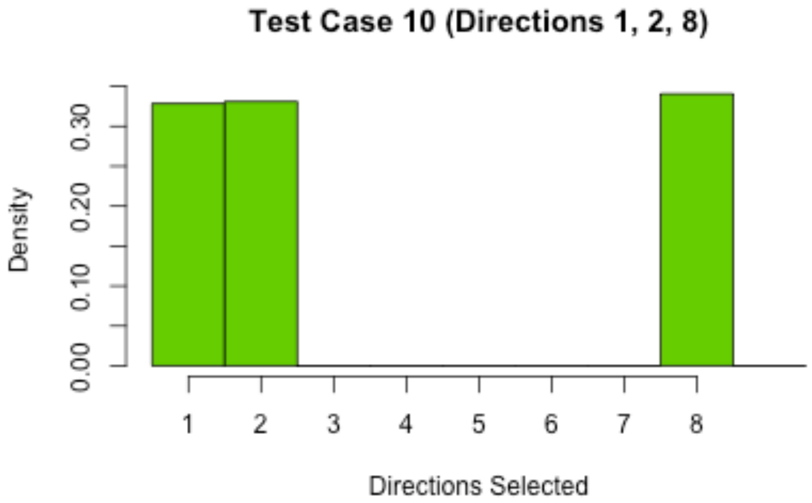


Figure 19: Agent Movement Validation Test Case 10.

Test Case 11

Row	35
Column	35
Previous Dir	1

1	1	1
0	x	0
1	0	1

Node 4	0.500
Node 5	0.952
Node 6	0.050

Oriented Movement, 5 resources available, randomly select from direction 1, 2, 4, 6, 8

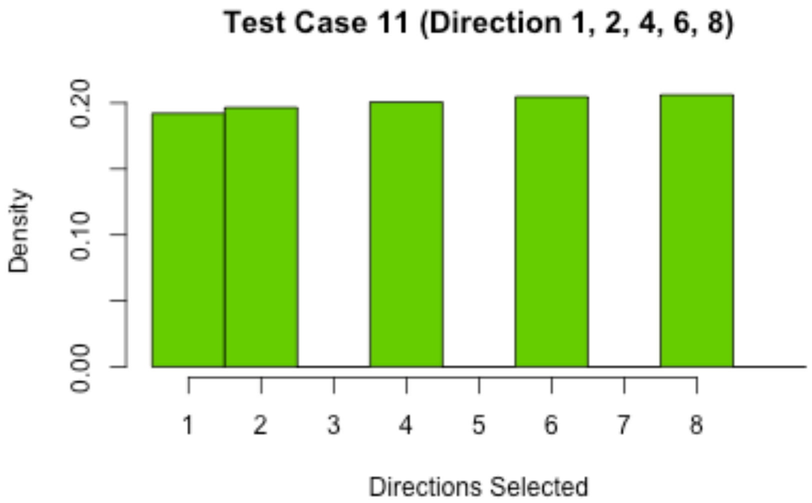


Figure 20: Agent Movement Validation Test Case 11.

Test Case 12

Row	35
Column	35
Previous Dir	1

1	1	1
1	x	1
1	0	1

Node 4	0.500
Node 5	0.952
Node 6	0.050

Oriented Movement, 7 resources available, randomly select from directions 1, 2, 3, 4, 6, 7, 8

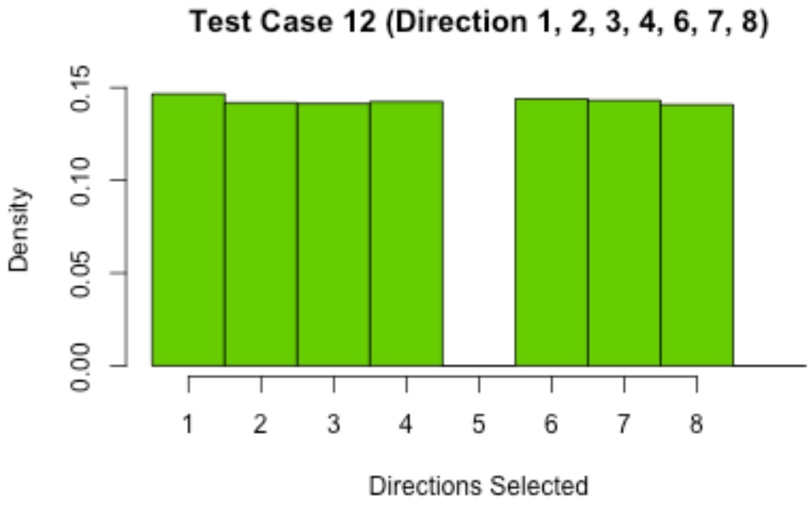


Figure 21: Agent Movement Validation Test Case 12.

6.1.3 Reproduction

The reproduction can be validated with edge cases. If the mutation and crossover are set to zero, the overall efficiency, defined as the ratio of resources gathered to time steps taken, should have a sharp initial increase (where the most efficient agents are copied perfectly) and then be relatively constant throughout all generations. As seen in Figure 22, below, this is the case when $P_{mut} = 0$ and $P_{xover} = 0$ (green line) with a constant efficiency of 20% after about 20 generations, where the variance is due to random placement of agents. When $P_{mut} = 0.2$ and $P_{xover} = 0.02$, the efficiency is increasing (blue line), with an efficiency of about 28% after G =300 and continuing to increase.

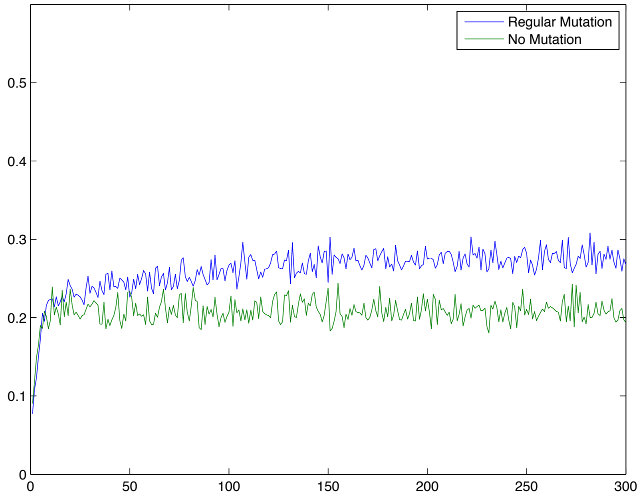
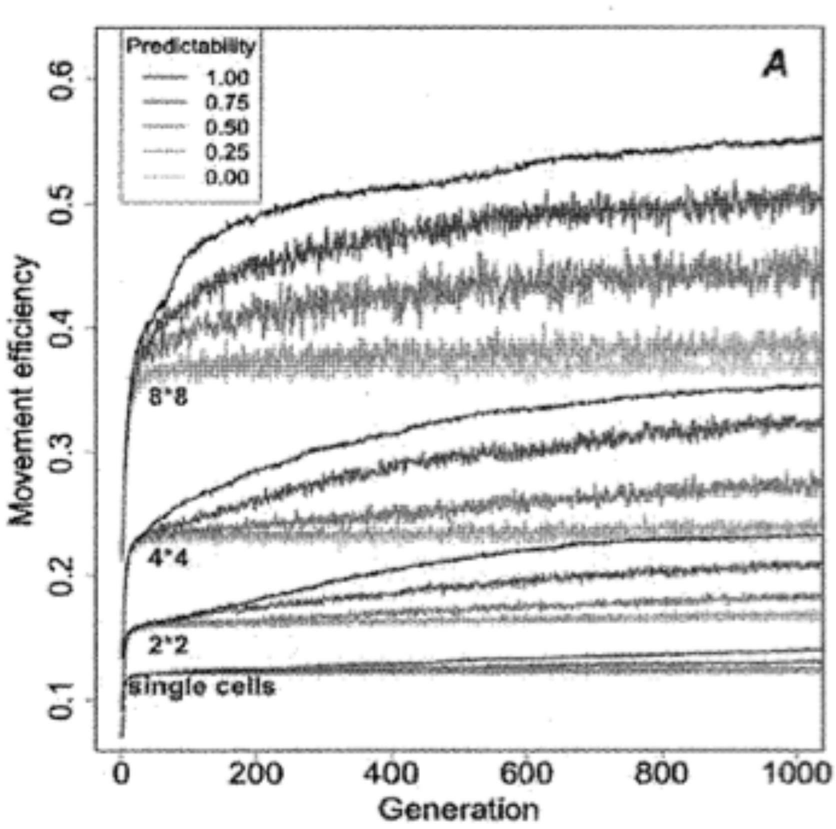


Figure 22: Reproduction Validation using edge case (blue) compared to model values for mutation and crossover (green).

6.1.4 Overall Stage 1

The overall model for stage 0 and 1 can be compared to an existing C implementation [5]. While many steps in the algorithm require random number generation, the overall trends in efficiency can be examined, where efficiency is defined as the average resources gathered per time step of the reproducing population. The results can be seen in Figure 23, below.


Predictability
1.00
0.75
0.50
0.25
0.00
A
Movement efficiency
0.1
0.2
0.3
0.4
0.5
0.6
8*8
4*4
2*2
single cells
0
200
400
600
800
1000
Generation

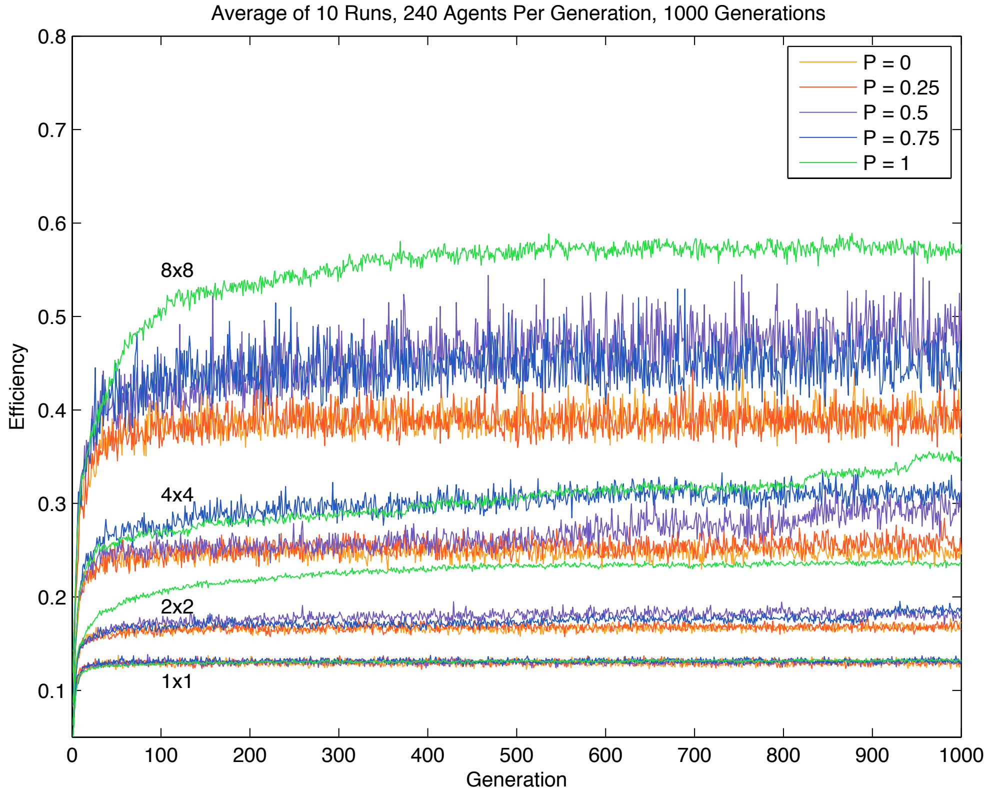

Average of 10 Runs, 240 Agents Per Generation, 1000 Generations
P = 0
P = 0.25
P = 0.5
P = 0.75
P = 1
Efficiency
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
8x8
4x4
2x2
1x1
0
100
200
300
400
500
600
700
800
900
1000
Generation

Figure 23: Validation of Stage 1 by comparison with results from C implementation. The top graph shows the efficiency (resources encountered per time step of the reproducing population) vs. generation, averaged over 100 runs for the C implementation by Mueller [5] (used with permission). The bottom graph shows the efficiency vs. generation, averaged over 10 runs for the MATLAB implementation. Trends and efficiency at G = 1000 are similar to C implementation, validating Stage 1 as a whole.

6.1.5 Overall Stage 2

As stage 2 is only a slight modification of stage 1, stage 2 can be further validated by programmatically checking that the agents do not occupy the same cell at the same time and by adding a few test cases concerning agent movement when other agents block neighboring cells. While the first produces no interesting graphical result, the test cases used can be seen in Figure 24 - Figure 26, below. In the following figures, the location of the agent, previous direction, calculated node values, and an image of the 8-cell neighborhood are given. The x represents the agent's location, the grey shaded cells represent that the map does not exist in those cells, 0 represents no resource in that cell, 1 represents a resource is present in that cell, and an A represents another agent is present in that cell.

Test Case 1h

Row	1
Column	64
Previous Dir	2

▨	▨	▨
A	x	▨
0	0	▨

Node 4	0.810
Node 5	0.906
Node 6	0.004

Memory Movement, direction of 1, but unavailable, and direction 7 is occupied, so randomly select from directions 6 or 7.

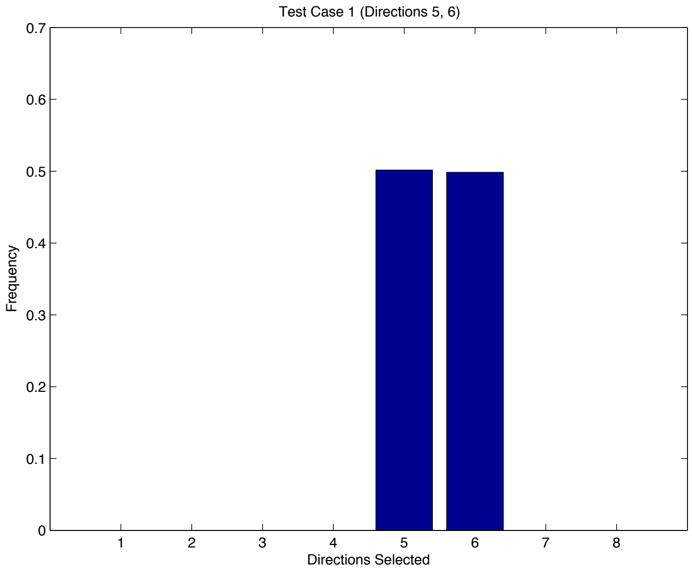


Figure 24: Agent Movement Validation for Herd Movement, Test Case 1h.

Test Case 2h

Row	64
Column	1
Previous Dir	6

▨	A	0
▨	x	0
▨	▨	▨

Node 4	0.981
Node 5	0.003
Node 6	0.053

Memory Movement, direction of 1, but occupied by agent, so randomly select from 2, 3.

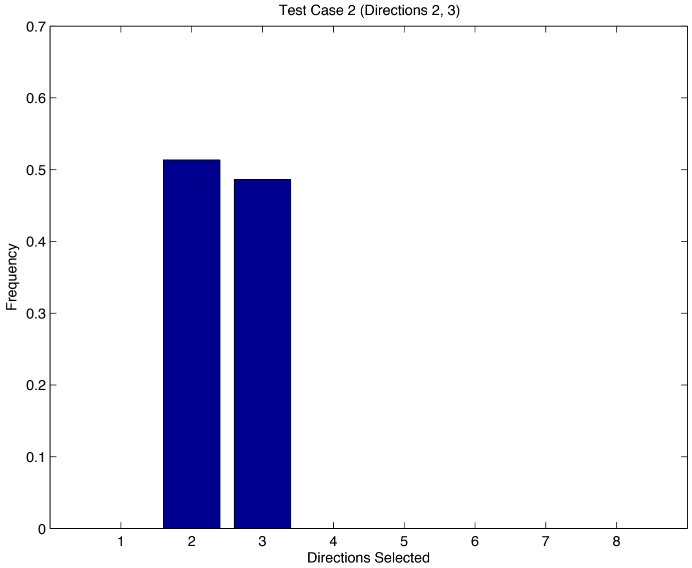


Figure 25: Agent Movement Validation for Herd Movement, Test Case 2h.

Test Case 3h

Row	64
Column	64
Previous Dir	4

A	A	▨
A	x	▨
▨	▨	▨

Node 4	0.501
Node 5	0.501
Node 6	0.490

Oriented Movement, but all surrounding cells occupied. No movement.

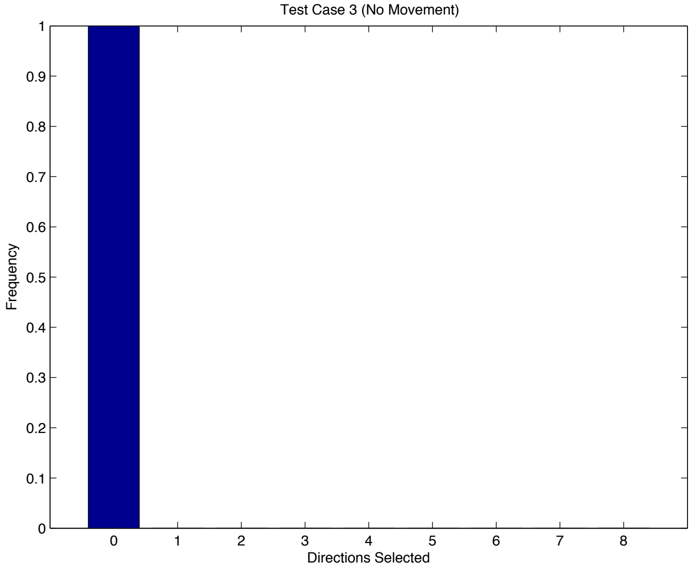


Figure 26: Agent Movement Validation for Herd Movement, Test Case 3h.

6.2 Spatial Statistics

The spatial statistic functions can be validated by comparison with existing implementations and/or hand calculations based on a small set of data points.

6.2.1 Realized Mobility Index

Finding the RMI involves two steps – first, finding the minimum convex polygon, and second, computing the area of this polygon. The minimum convex polygon found by the RMI function can be compared to the results of the R function `chull`, [8] which will compute the minimum convex polygon via a different algorithm, but should yield the same result. For validation, varying numbers of points were selected from a random uniform distribution to generate data sets. The hull created by the RMI function was compared, point by point, to the hull generated by the `chull` function. The number of points of the hull that were not the same (mis-matched points) were recorded and shown in Figure 27, below. In all cases, the hull generated by `chull` and the hull generated by RMI were exactly the same.

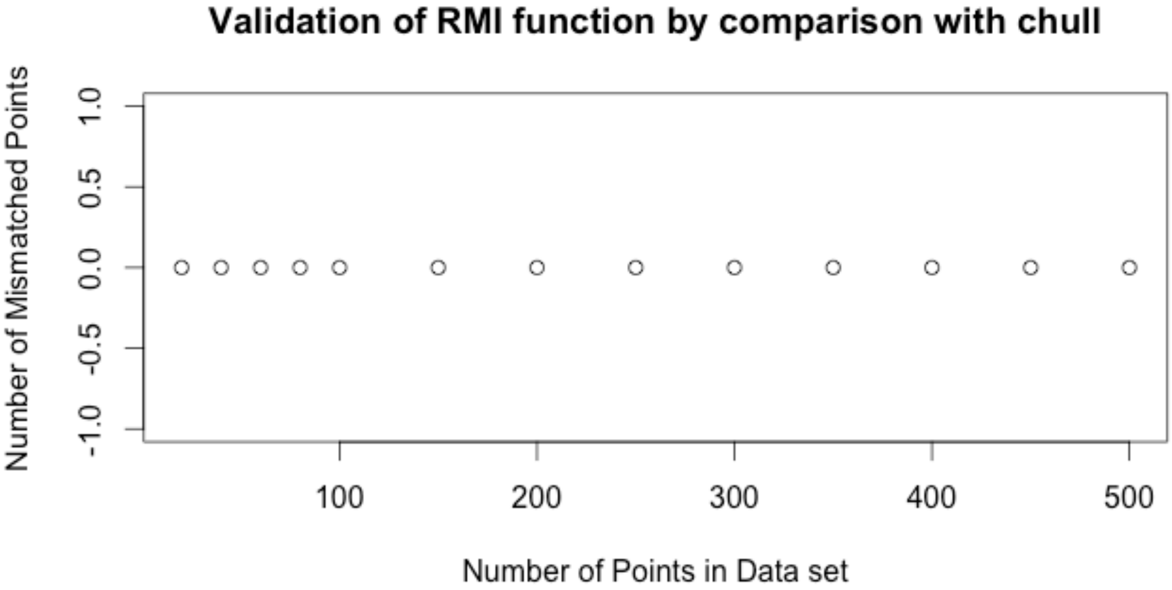


Figure 27: Validation of RMI Function by comparison with chull function.

The area calculation of a convex polygon that occurs in the RMI function was validated by hand computation of simple convex polygons.

6.2.2 PDI

The PDI function can be validated via comparison to the R function `k12hat`, from the SPLANCS library [9]. For the validation of the PDI function via the `k12hat` function, the following algorithm was used.

1. Randomly select two numbers (with replacement) from {10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 150, 200} to be the number of data points in set 1 and the number of data points in set 2.

2. Randomly select a number from {10, 20, 30, 40, 50} to use as max coordinate value. Use random uniform distribution to select data points for set 1 and set 2 in the range of [-max coordinate, max coordinate].
3. Determine rectangular hull of the combined set of points.
4. Randomly select a value of S from a uniform distribution of [0, 0.5(Area of rectangular hull)$^{0.5}$]
5. Compute PDI using both k12hat and PDI function. Calculate error of PDI function relative to k12hat function.

The resulting plot can be seen in Figure 28, below. The relative error is very small, validating the PDI function.

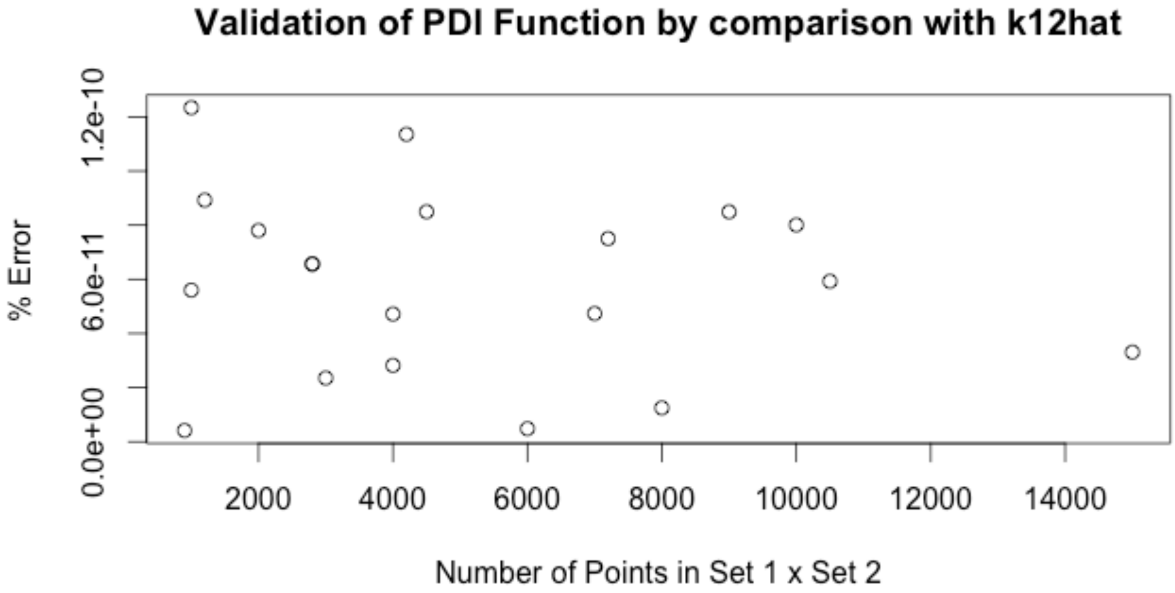


Figure 28: Validation of PDI by comparison with k12hat. The relative error for all sizes of sets tested was below 10^{-10}.

6.2.3 MCI

As the MCI metric is simple computationally (around 10 lines) it can be validated by a small hand computation.

7. Testing

7.1 Timing – Agent Based Model

Using the R profiler, the self-times for each function can be obtained for the implementation of the agent based model in R, as shown in Figure 29, below.

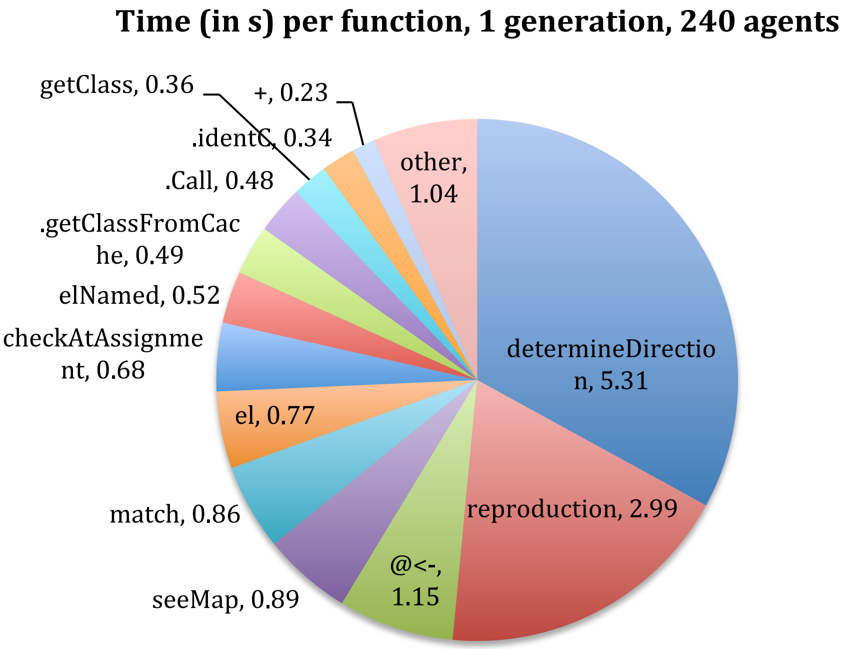


Figure 29: Self time, in seconds, per function

With the exception of the landscape regeneration functions, which are included in the 'other' category because they took so little time, all functions depend on number of agents. Thus, these times depend linearly on *N*, the number of agents in each generation. Determining direction takes the most amount of time per generation, but as this function is mostly conditional statements in order to match the agent movement algorithm, it is not likely that the time spent in this function can be reduced. However, other functions, such as `checkAtAssignment`, `elNamed`, `el`, and match, are all internal R functions that are called when data elements in user-defined classes are accessed. The total time for one generation in the R implementation is 16.11 seconds, which was too slow to make it feasible to run all 20 combinations of landscape variability and patch size.

The agent-based model was then implemented in MATLAB. The time per generation in the MATLAB implementation was 2.04 seconds, which allowed for all 20 combinations of landscape variability and patch size to be run for 1000 generations in around 11 hours and 20 minutes. Using the MATLAB profiler, the self-times for each function can be obtained, as shown in Figure 30 below. The 'mysim' function includes both reproduction and refreshing the landscape. Again, the function that determines the direction takes the most time, as it must evaluate many conditional statements.

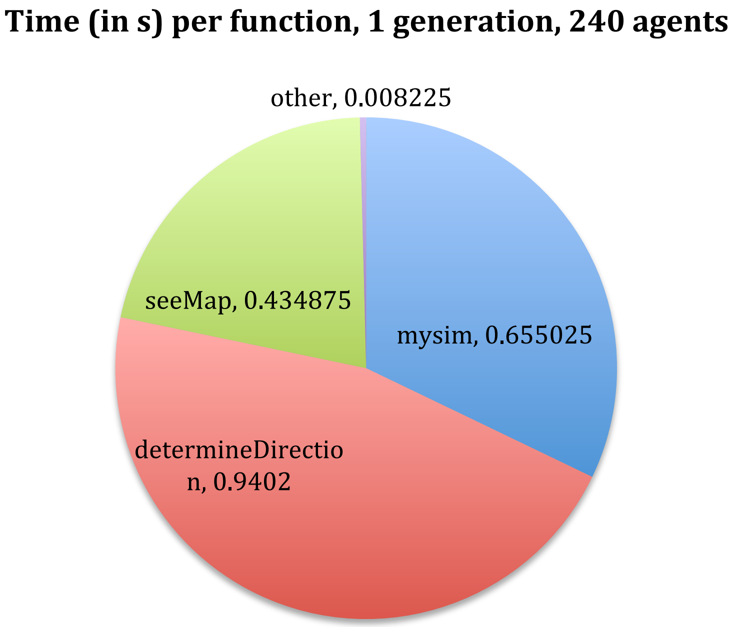


Figure 30: Self time, in seconds, per function for MATLAB implementation

7.2 Timing – Spatial Statistics

The PDI function was initially implemented fully in R. However, for one set of the agent based model simulation data, there would be a total of 12,420,000 point comparisons, which would be prohibitively slow. Thus, the function was implemented mostly in C, and the C function is called from within an R wrapper. The version of the PDI function that uses the C function is approximately 6000 times faster than the version that is entirely implemented in R. For timing testing, two randomly selected sets of points from the uniform distribution were chosen and the two PDI functions were used to compute PDI. These two PDI values were checked to ensure they were the same. A plot of a sample timing run can be seen in Figure 31, below.

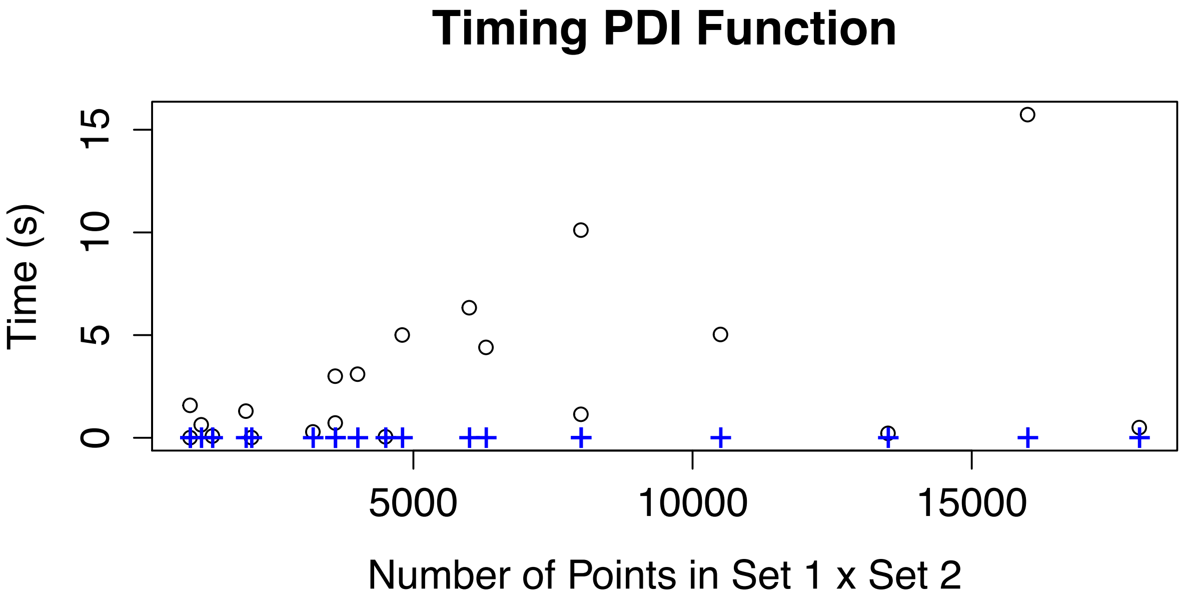


Figure 31: Timing for the two implementations of the PDI function. The black points (circles) are from the function implemented entirely in R, and the blue points (+) are from the function implemented in C, then wrapped in an R function.

The timing of the convex-hull-finding part of the RMI function was tested by randomly generating varying sizes of data points from a uniform distribution and then timing how long it took to find the convex hull. The number of points needed to make the hull was recorded. Theoretically, according to [2], the time should go as nlog(H), where H is the number of points on the hull, and n is the total number of points. This is confirmed as when nlog(H) is plotted against the time for finding the hull, it is linear, as seen in Figure 32, below.

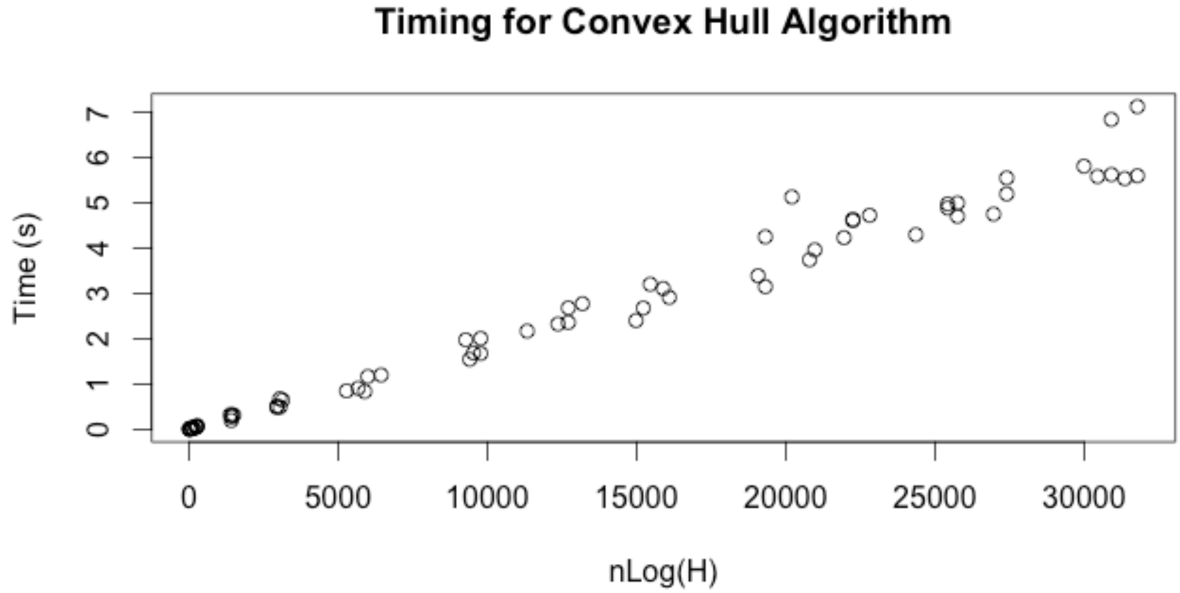
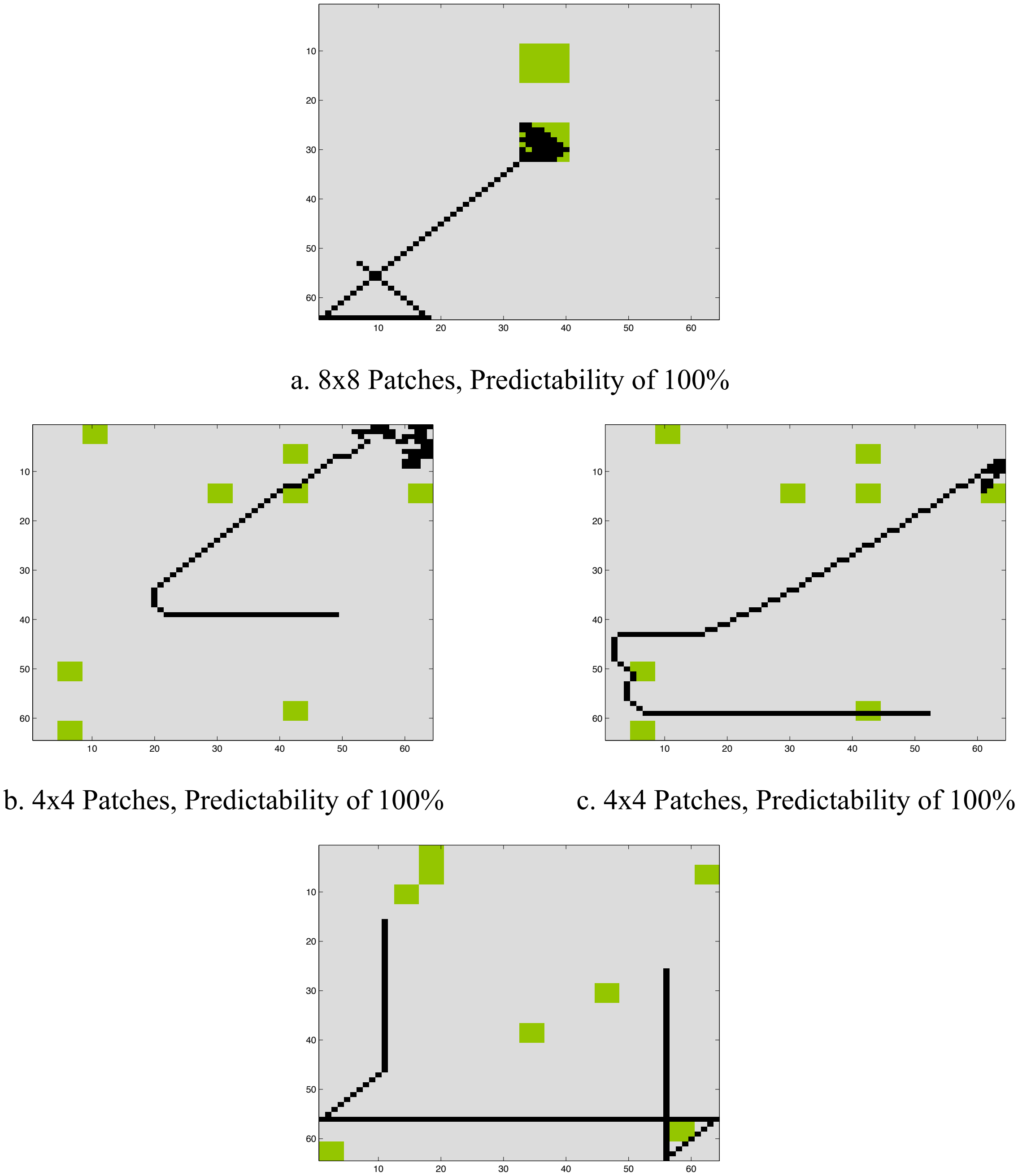


Figure 32: Timing for the convex hull finding part of the RMI algorithm

7.3 Testing with Simulation Data

Testing of the RMI, PDI, and MCI was done using the gazelle data set and the Stage 2 simulation data sets from all 20 different landscape combinations. Additionally, simulation data sets at a lower temporal sampling rate were created by selecting every other point, every 5th point, and every 10th point from the full sets of simulation data. RMI, PDI, and MCI statistics for the combinations of landscape predictably and patch size will be examined to see if these statistics are sufficient for determining large-scale population patterns. Intuitively, one might expect the following for the different population patterns. For home-range behavior, a small RMI would be expected, as individuals stay within their home range, which is likely a small portion of the entire species range. A small PDI at all spatial scales is expected as individuals would stay in their own area, and a small or negative MCI at all time scales is also expected as the individuals would not interact with each other. For a migratory population, the RMI would be large, as the individuals would be moving over most of the range of the species as they migrated from one location to another. The PDI is expected to be large and increasing with increasing distance, as the individuals would move together at roughly the same time, making the points of the individual clustered with the points of other individuals. The MCI is expected to be low at high sampling rates, but high at lower sampling rates, as in a large time scale the individuals should be moving together. For a nomadic population, the RMI, MCI, and PDI would likely be in between that of a home range population and a migratory population. Sample tracks for different landscape parameters are shown in Figure 33, below.



a. 8x8 Patches, Predictability of 100%

b. 4x4 Patches, Predictability of 100%

c. 4x4 Patches, Predictability of 100%

d. b. 4x4 Patches, Predictability of 0%

Figure 33: Example Tracks for Various Landscape Parameters

7.3.1 Testing with Simulation Data – RMI

The RMI for all simulation data sets can be seen in Figure 34, below. In general, the RMI decreases slightly as the sampling rate is decreased, which makes sense as the full movement of the individuals may not be captured by a less frequent sampling of position.

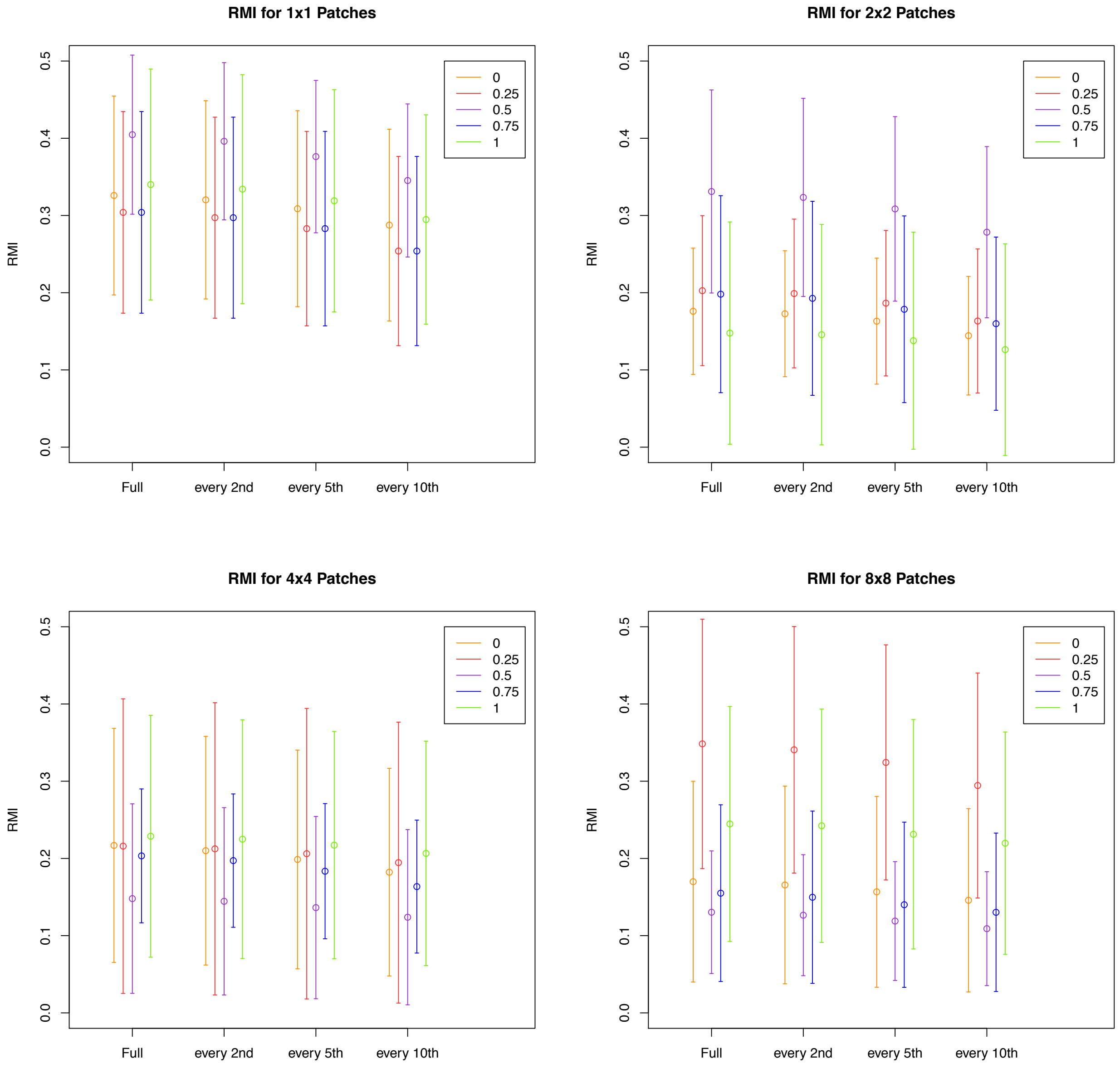


Figure 34: The RMI for all simulation data sets. Different colors represent different levels of landscape predictability. For all data sets, a lower sampling rate (every 2nd, every 5th, every 10th point) leads to lower RMI relative to the full data set.

For a sense of RMI trends across the differing landscape combinations, Figure 35 shows a summary of RMI for the full simulation data set. The largest RMI values are from a patch size of 1x1, with generally smaller RMI values as the patch size increases.

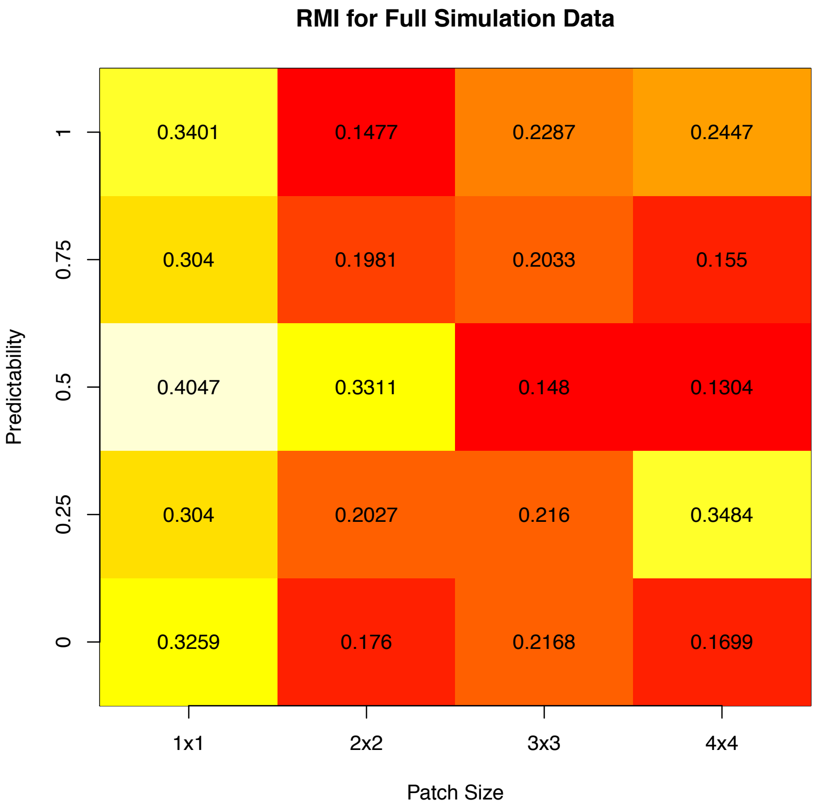
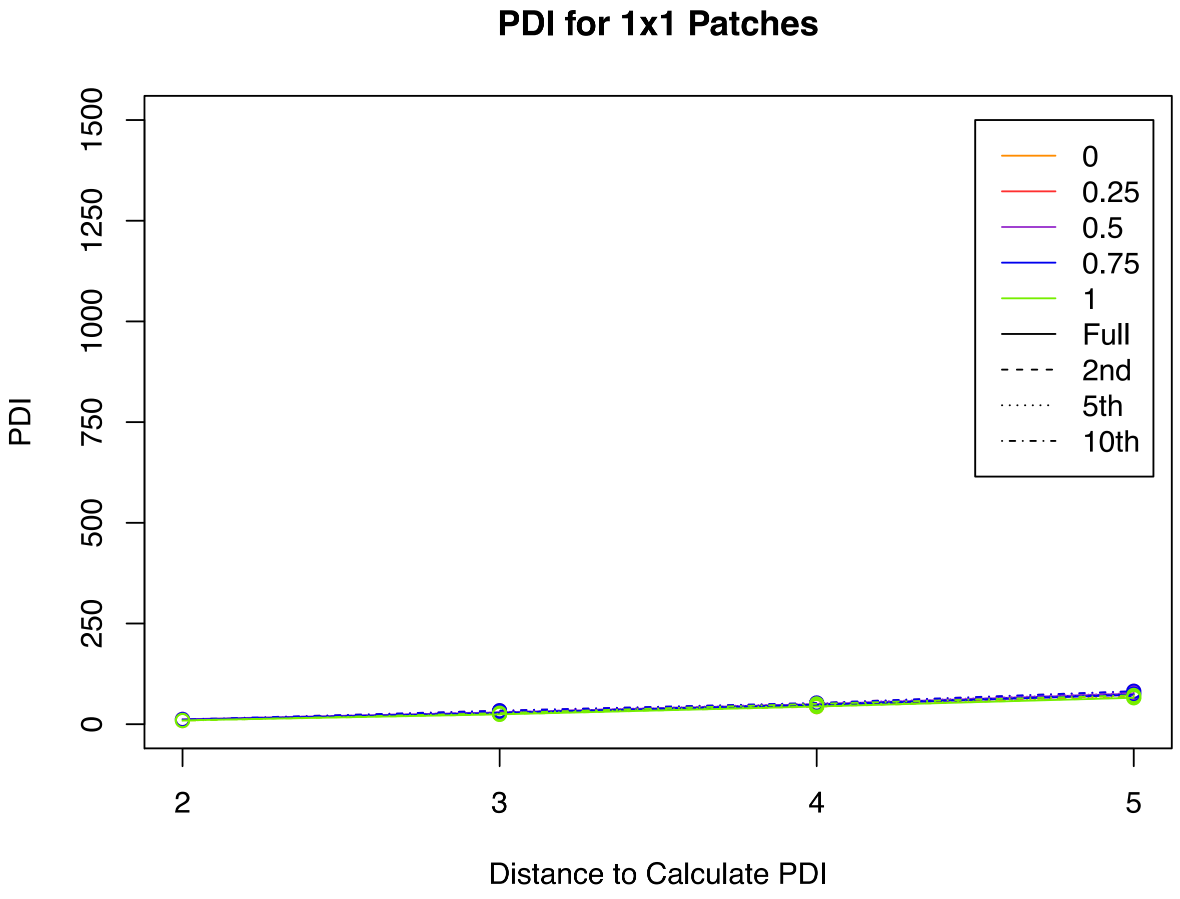
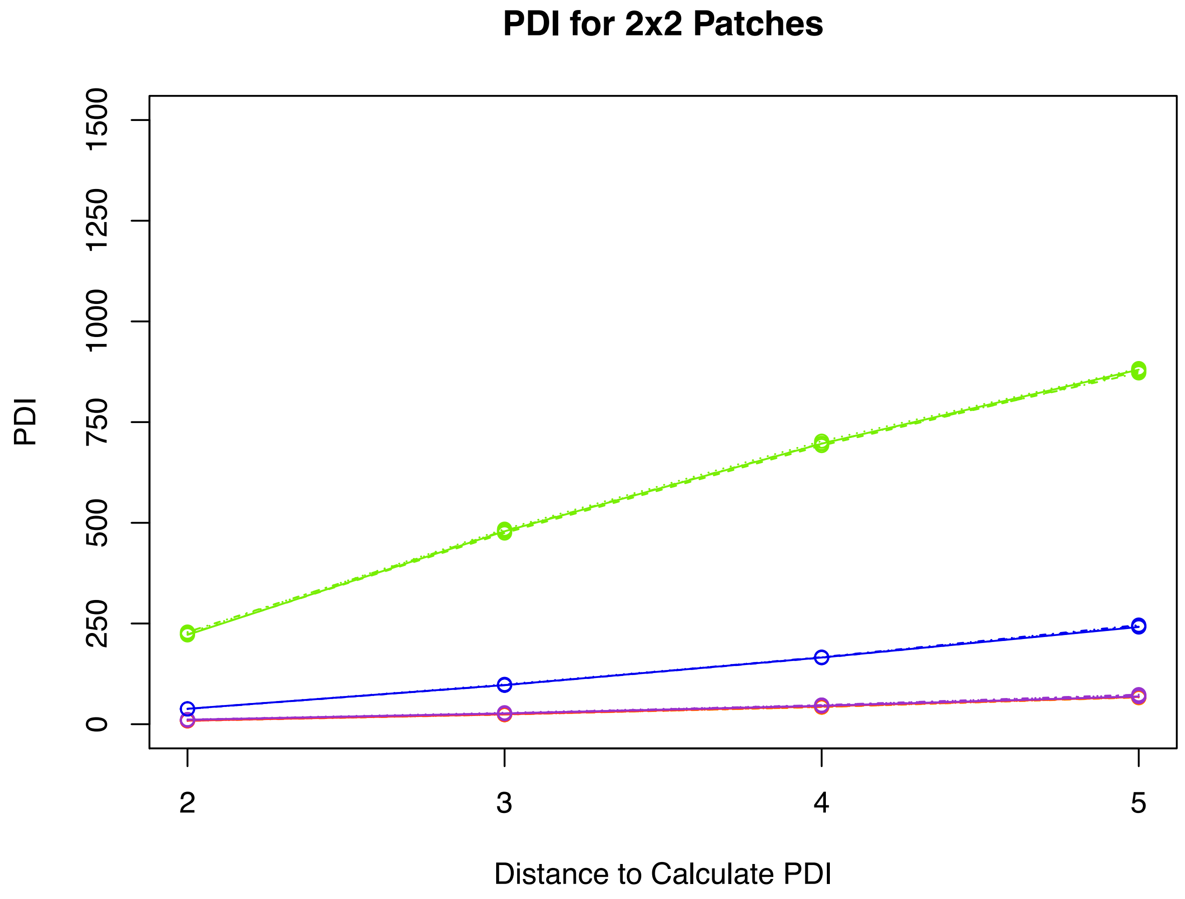


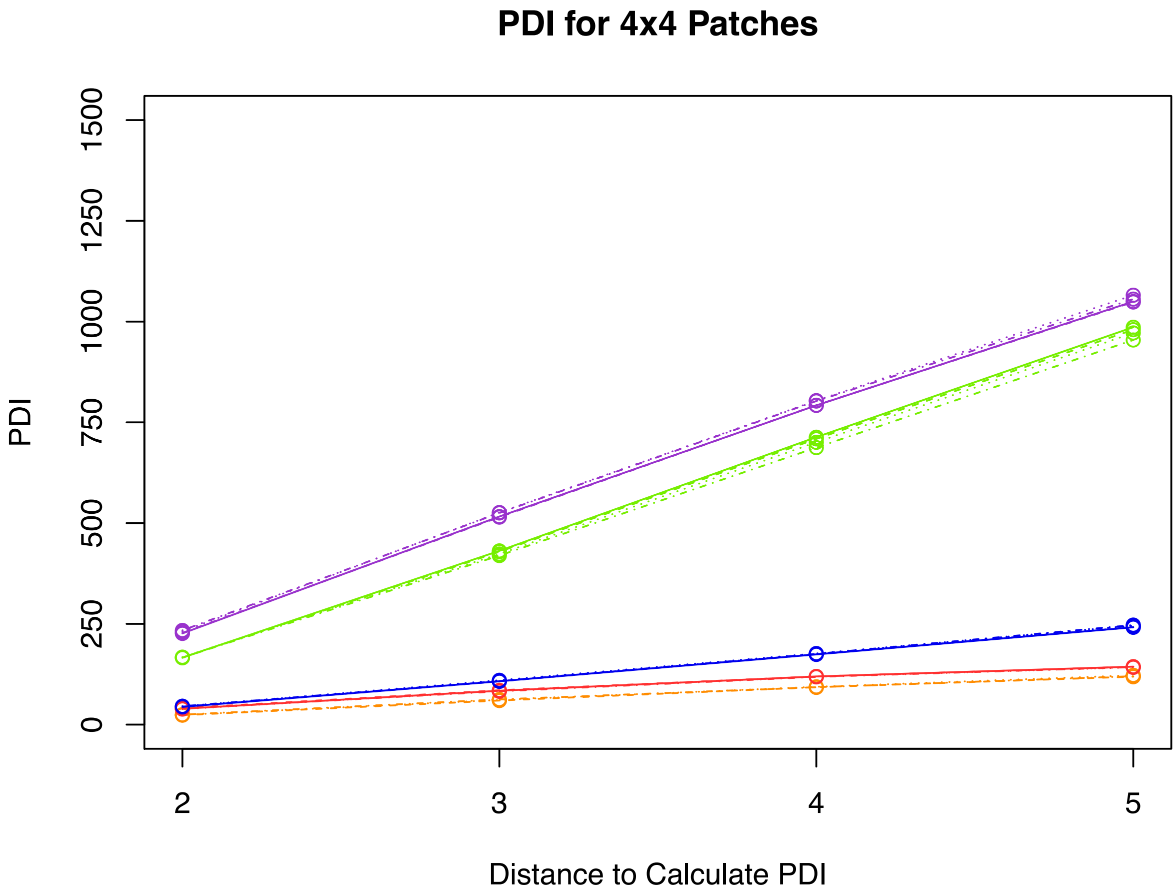
Figure 35: RMI for Full Simulation Data Set

7.3.2 Testing with Simulation Data – PDI

The PDI for the full sets of simulation data can be seen in Figure 36, below. The PDI changes only slightly with differing sample rate. As the patch size increases, the PDI for predictable landscapes increases, indicating greater clustering of individuals when the landscape is predictable. This is sensible as the individuals would have evolved to locate the resources at specific (predictable) locations, and thus would be clustered around these resources. As the patch size increases, there are fewer total patches, and thus more clustering would occur. Even at large patch sizes, the clustering is low for unpredictable landscapes as the individuals are largely engaged in random walks as they search for resources.


PDI for 1x1 Patches
PDI
0
250
500
750
1000
1250
1500
Distance to Calculate PDI
2
3
4
5
0
0.25
0.5
0.75
1
Full
2nd
5th
10th


PDI for 2x2 Patches
PDI
0
250
500
750
1000
1250
1500
2
3
4
5
Distance to Calculate PDI


PDI for 4x4 Patches
PDI
0
250
500
750
1000
1250
1500
2
3
4
5
Distance to Calculate PDI

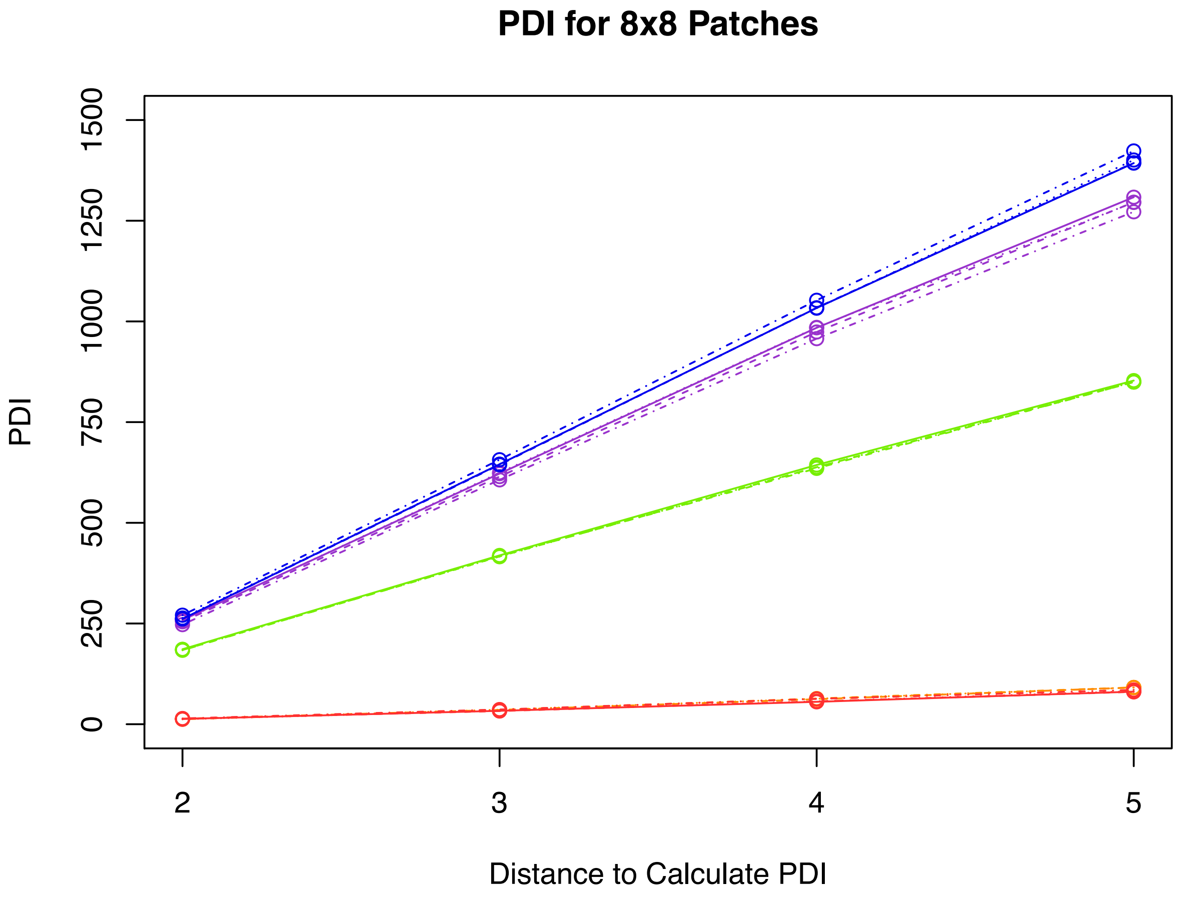


Figure 36: PDI For all simulation data sets, grouped by patch size. Colors indicate predictability, and line type indicates sample rate (every point, every 2nd point, every 5th point, every 10th point). The PDI does not change considerably for different sample rates. The PDI increases as patch size increases.

7.3.3 Testing with Simulation Data - MCI

The MCI for the full sets of simulation data can be seen in Figure 37, below. For low predictability (Pr = 0%, 25%) and small patch sizes (r = 1), the correlation between the individuals is near zero. For the full data sets, the correlation between individuals at larger patch sizes with high predictability is slightly negative. This is likely because the individuals cannot occupy the same cell, and as they are tightly clustered, must move away from each other to move at all. Somewhat surprisingly, this negative correlation increases as the sampling rate decreases. This indicates that the individuals are moving away from each other on a larger time scale. This may be due to the random initial placement of the individuals. The individuals placed close to a resource patch initially will get to the patch quickly, and once the patch is depleted, move away from the patch. The individuals that are further from a patch will still move toward the patch without being aware that the patch has been depleted, leading to many individuals moving toward the patch while others are moving away from the patch.

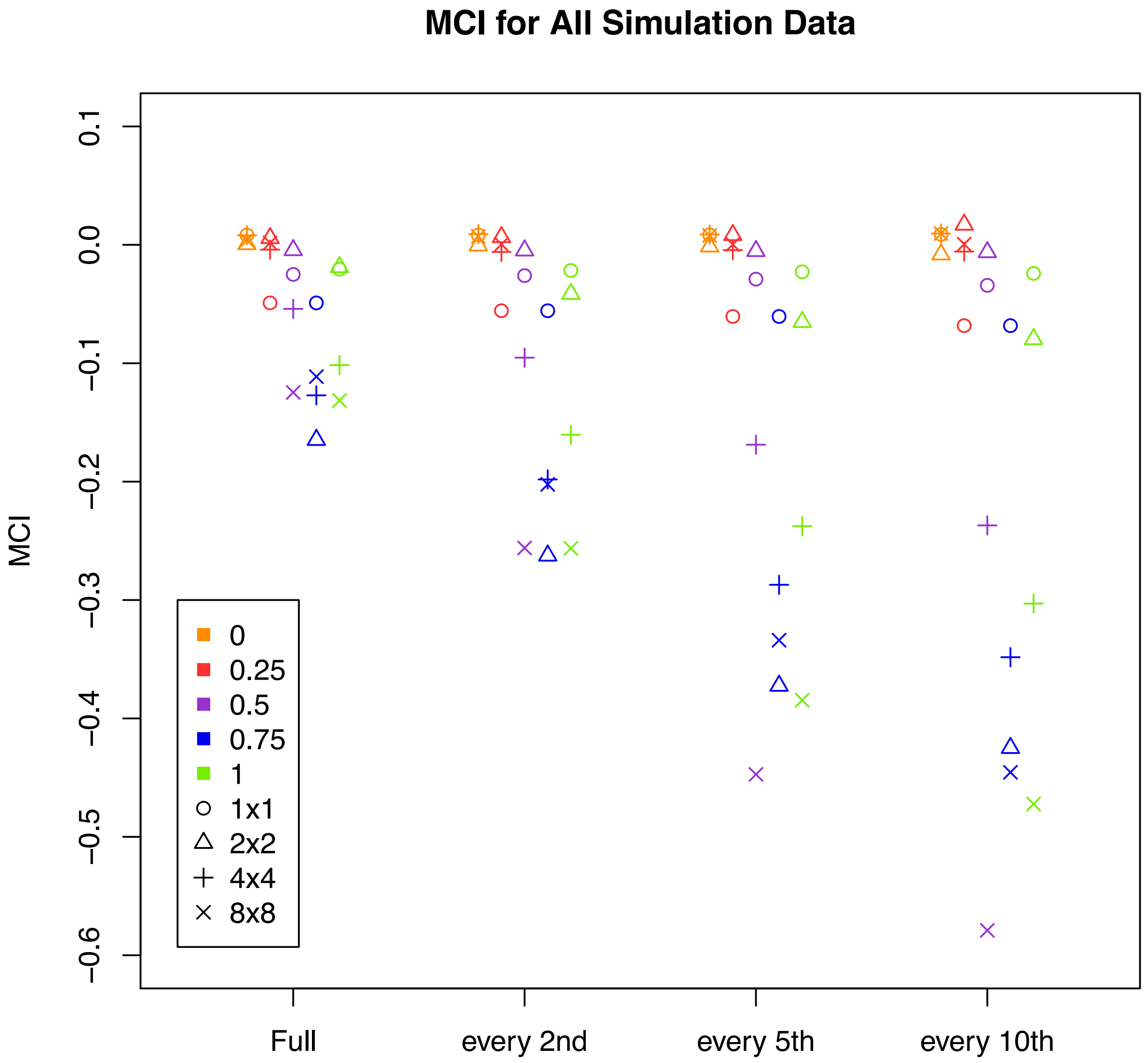


Figure 37: MCI for all simulation data. As the sampling rate decreases to every 10th day, the more predictable landscapes with larger patch sizes exhibit large negative correlation.

7.4 Testing with Gazelle Data

The spatial metrics for the gazelle data are summarized in Table 2, below. As the absolute positions do not affect the metrics, the gazelle data was preprocessed to give a similar sample area as the simulation data. For reference, an S of 2 corresponds to approximately 6 km. The RMI and PDI metrics do not depend on time, and thus the full set of data could be used to compute these metrics. However, the MCI requires relocation data that is taken at the same time for all individuals with equal time steps. The MCI was calculated for a reduced set of 13 individuals at a sample rate of 24 hours, as that was the sample rate and individual subset that allowed for the largest data set that could be used for the MCI. The gazelle have a PDI and an MCI that is consistent with the smaller patch sizes or unpredictable landscapes, and a much smaller RMI than any of the simulation data sets. The smaller RMI is likely due to the large time span over which the data was recorded as well as the fact that the gazelles did not have the same restrictions on movement as the simulated individuals.

Table 2: Spatial Metrics for Gazelle Data (scaled to match simulation area)

MCI	-0.1273
RMI	0.0751±0.0676
PDI (S=2)	29.99±14.52
PDI (S=3)	60.18±26.30
PDI (S=4)	97.41±40.61
PDI (S=5)	140.39±56.10

8. Concluding Remarks

8.1 Project Summary

While initially R was chosen as the implementation language because it is familiar to biologists, it ultimately was not fast enough for the algorithms in this project because of the prevalence of for loops. Re-implementing the agent based algorithm in MATLAB made the run time about 8 times faster than the R implementation. Implementing PDI in C and adding an R wrapper made the run time almost 6000 times faster than the R implementation, and because it was wrapped in an R function, it still can be easily integrated with other metrics and used by biologists.

While the initial goal was to determine if the three spatial metrics could discern between the different types of population level movement, it is not likely that the simulation was sufficiently realistic to allow for the typical population level movements (home range, migratory, nomadic) to emerge. While the behavior of the individuals as well as the spatial metrics did vary with varying resource patch size and temporal variability, the mechanisms that may be required for some of the types of population level patterns to emerge were not present. For example, migratory behavior would likely require a landscape that had periodic generation of resources in at least two spatially distinct areas. As the resources did not regenerate in this model, it is unlikely that periodic movement between resource patch areas would occur. Despite the lack of typical population patterns emerging, the generation of simulation data did allow for the examination of the metrics when the data was "collected" at different sampling rates. Specifically, while the PDI metric did not change appreciably with a differing sample rate, and the RMI predictably decreased with a lower sample rate, the MCI changed greatly with a differing sample rate for larger and more predictable patches. This makes sense as the overall correlation of individual movements may only be present at larger time scales. For example, one might expect an hourly sampling of a migratory population to not be correlated, but a weekly sampling during a migration to be highly positively correlated.

8.2 Future Areas of Research

The large negative correlation between individuals when the patch size is large and the landscape is highly predictable is somewhat surprising, but may be due to the random placement of individuals across the entire simulation area. It may be interesting to try placing them in a similar starting location to determine if this affects the MCI metric. Aside from not being allowed to occupy the same cell, the individuals in the herd simulation had no awareness of the other individuals. While all individuals had the same “memory” of the patch locations as well as the same “genetic material”, this did not appear to be sufficient to generate desired population level patterns. Future research could include some attractive or repulsive force between different individuals, as well as allowing the landscape to regenerate resources according to specific schedule in order to model temporal resource availability in a real world landscape.

9. Milestones and Deliverables

The project schedule as initially proposed is shown below in Table 3. Milestones coincide with the bolded elements in the project schedule.

Table 3: Project Schedule and Milestones

Week (Planned)	**Tasks**
October 21 – November 3	Set up data structures for agents ✓ Implement landscape initialization ✓ Implement landscape update ✓ Implement agent initialization ✓
November 4 - November 17	Implement agent direction ✓ Validate agent direction ✓
November 18 – December 1	Validate landscape initialization ✓ Implement reproduction ✓ Validate reproduction ✓
December 2 – December 16	**Connect implementations for full stage 0 + stage 1 ✓** Create mid-year report✓ **Give mid-year presentation ✓**
January 20 – February 2	**Implement Stage 2✓**
February 3 – February 16	**Implement RMI✓**
February 17 – March 2	**Implement PDI✓** **Implement MCI✓**
March 3 – March 16	Validate PDI, RMI, and MCI✓
March 24 – April 6	Validate Stage 0, Stage 1, Stage 2✓
April 7 – April 20	Test RMI and PDI with gazelle data✓ Test RMI and PDI with full simulation data✓ Test RMI and PDI with degraded simulation data✓
April 21 – May 4	Write final report and create final presentation✓
May 5 – May 19	Complete final report and presentation✓ **Give final presentation, submit final report✓**

Deliverables for this project are:

1. Proposal document
2. Proposal presentation
3. Documented code for agent based model
4. Documented code for RMI, PDI, MCI (with examples for use)
5. Data sets created by simulation (degraded and full)

6. Gazelle data set
7. Mid-year report
8. Mid-year presentation
9. Final report
10. Final presentation

Bibliography

1. Calabrese, Justin, Chris H. Fleming, Bill F. Fagan, Marin Rimmler, Petra Kaczensky, Peter Leimgruber, and Thomas Mueller. From the fish tank to the Gobi desert: A general, scalable approach to quantifying animal movement coordination. (Unpublished, 2013)
2. Cressie, Noel A. *Statistics for spatial data*. New York: Wiley, 1993
3. Kirkpatrick, David G., and Raimund Seidel. "The ultimate planar convex hull algorithm?." *SIAM journal on computing* 15.1 (1986): 287-299.
4. Lotwick, H. W., and B. W. Silverman. "Methods for analysing spatial processes of several types of points." *Journal of the Royal Statistical Society. Series B (Methodological)* (1982): 406-413.
5. Mueller, Thomas, William F. Fagan, and Volker Grimm. "Integrating individual search and navigation behaviors in mechanistic movement models."*Theoretical Ecology* 4.3 (2011): 341-355.
6. Mueller, Thomas, and William F. Fagan. "Search and navigation in dynamic environments–from individual behaviors to population distributions." *Oikos*117.5 (2008): 654-664.
7. Mueller, Thomas, et al. "How landscape dynamics link individual‐to population‐level movement patterns: a multispecies comparison of ungulate relocation data." Global Ecology and Biogeography 20.5 (2011): 683-694.
8. R Core Team (2013). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.
9. Rowlingson, Barry and Peter Diggle (2013). splancs: Spatial and Space-Time Point Pattern Analysis. R package version 2.01-34. http://CRAN.R-project.org/package=splancs

Appendix - Algorithms

1. Agent Based Model

1.1 Overall Model

The following algorithm describes how the overall model was implemented. For ease of reading, the landscape initialization, landscape updates, agent direction selection, and reproduction algorithms are separated and detailed following the main algorithm.

For each execution of this overall algorithm, a patch size (*r*) and predictability (*Pr*) will be chosen.

```
Stage 0 – Initialization
    1. For i = 1:N
        a. Select 33 random integers [-5, 5] to be initial weights for the neural network for
           agent i.
    2. Initialize landscape

Stage 1 – Evolution
    1. For generation j = 1:G
        a. For agent i = 1:N
            i. Place agent i in a random starting location.
            ii. For t = 1:150
                1. Determine direction to move.
                2. Move to new location.
                3. If resource exists at that location, add one to resource counter for
                   agent, subtract one from resource counter for cell.
            iii. Renew landscape to state at beginning of generation.
        b. If j != G
            i. Create N new agents according to reproduction algorithm.

Stage 2 – Herd Movement
    1. Choose agent from last generation with largest resource counter and create 24 copies with
       no mutation
    2. For t = 1:150
        a. For j =1:N
            i. Choose without replacement an agent to update.
            ii. If t = 1
                1. Place agent in random starting location. If agent already exists at
                   that location, select a new location.
            iii. Else (t > 1)
                1. Determine direction to move.
                2. Move to new location.
                3. If resource exists at that location, add one to resource counter for
                   agent, subtract one from resource counter for cell.
                4. Record agent location.
```

1.2 Landscape Initialization

```
1. For i:$R/r^2$
    a. Randomly select a location for the top left corner of patch i.
    b. If the patch does not overlap another patch or the edge of the grid
        i. Place patch i (set the value of cells in patch i to 1 for evolution stage or N
           for herd movement stage).
    c. Else, go to Step 1a.
2. Select $\frac{R}{r^2} \times Pr$ patches to remain constant throughout the generations. Let $C$ be the set of
   patches that will remain constant.
```

1.3 Landscape Update

```
1. Start with blank landscape.
2. Place all patches in $C$.
3. For i:# of patches not in $C$
    a. Randomly select a location for the top left corner of patch i.
    b. If the patch does not overlap another patch or the edge of the grid
        i. Place patch i (set the value of cells in patch i to 1 if evolution stage
           generation counter is $<G$, N if generation counter is $>G$)
    c. Else, go to Step 3a.
```

1.4 Agent Direction Selection

The selection of a movement direction of an agent proceeds according to the following algorithm. If at any time a cell is unavailable due to another agent occupying that cell or due to cells not existing off the edge of the grid, the agent will move in a random direction selected from the available directions.

```
1. Compute hidden layer node values and output layer node values according to 3.1.2.
2. If Node 4 is [0, 1/3), use non-oriented movement.
    a. If Node 5 is [0,1/3), move in same direction as previous time step, set search
       effort to 0.
    b. If Node 5 is [1/3,2/3), set search effort to 1 and randomly select between
          i. Same direction as previous time step.
         ii. 45 degrees CW from previous time step.
        iii. 45 degrees CCW from previous time step.
    c. Else, Node 5 is [2/3,1] set search effort to 9 and randomly select direction.
3. Else, if Node 4 is [1/3,2/3), use oriented movement. Set search effort to 0.
    a. If there is more than one resource in the surrounding cells, randomly select a cell
       with resource as new location.
    b. Else, if there is one resource, select that cell as new location.
    c. Else, there are no resources, randomly select cell.
4. Else, Node 4 is [2/3,1], set search effort to 0 and use memory-oriented movement.
    a. If Node 6 is [0,1/8), go N.
    b. If Node 6 is [1/8,2/8) go NE.
    c. If Node 6 is [2/8, 3/8), go E.
    d. If Node 6 is [3/8, 4/8), go SE.
    e. If Node 6 is [4/8, 5/8), go S.
    f. If Node 6 is [5/8, 6/8), go SW.
```

```
        g. If Node 6 is [6/8, 7/8), go W.
        h. If Node 6 is [7/8, 8/8), go NW.
```

1.5 Reproduction

```
    1. For i = 1:($N$/6)
        a. Randomly select six agents without replacement.
        b. Find agent with largest resource counter, call this agent A.
        c. For j = 1:6
            i. For k = 1:33
                1. Choose random number [0,1] from uniform distribution.
                2. If random number is <$Pr_{mut}$
                    a. Choose integer from uniform distribution [-5,5] and set this
                       value as gene k for agent j.
                3. Else, set gene k for agent j to the value of gene k for agent A.
    2. Choose $Pr_{xover} xN$ agents to cross over. Pair agents by randomly ordering the indices of
       selected agents.
    3. For i = 1:2:($Pr_{xover} xN$)
        a. Choose random integer [1,33] from uniform distribution; this is the start of the
           crossover, call it c.
        b. Swap genes c:33 for agent i with genes c:33 for agent i+1.
```

2. RMI

2.1 Algorithm for Finding Upper Hull

Note: in this algorithm, $x(p_i)$ represents the x value of the data point p, with index i. The set of all data points is P.

```
    1. Initialization – Let min and max be the indices of two points in A that form the left and
       right endpoints of the upper hull of P.
                         $x(p_{min}) \leq x(p_i) \leq x(p_{max})$
                     $y(p_{min}) \geq y(p_i)$ if $x(p_{min}) = x(p_i)$,
              $y(p_{max}) \geq y(p_i)$ if $x(p_{max}) = x(p_i)$ for $i = 1, \dots, n$
       If min = max, print min and stop.
       Let $T := \{p_{min,}p_{max}\} \cup \{p \in P | x(p_{min}) < x(p) < x(p_{max})\}$
    2. CONNECT(min, max, T)
```

2.2 Algorithm for CONNECT(k, m, P)

```
    1. Find a real number a such that
                 $x(p_i) \leq a$ for $\left\lceil \frac{|P|}{2} \right\rceil$ points in P and
                 $x(p_i) \geq a$ for $\left\lfloor \frac{|P|}{2} \right\rfloor$ points in P
    2. Find the "bridge" over the vertical line $L = \{(x, y) | x = a\}$
                              $(i, j) := BRIDGE(P, a)$
    3. Let $P_{left} := \{p_i\} \cup \{p \in P | x(p) < x(p_i)$
```

```
   Let $P_{right} := \{p_j\} \cup \{p \in P | x(p) > x(p_i)$
4. If i = k then print(*i*)
5. Else CONNECT(*k, i, $P_{left}$*)
6. If j = m then print (*j*)
7. Else CONNECT (*j, m, $P_{right}$*)
```

2.3 Algorithm for BRIDGE(P,a)

```
1. $CANDIDATES := \emptyset$
2. If |P| = 2 then return ((*I,J*)), where $P = \{p_i, p_j\}$ and $x(p_i) \le x(p_j)$.
3. Choose $\left\lfloor \frac{|P|}{2} \right\rfloor$ disjoint sets of size 2 from P, call these sets PAIRS.
4. Determine the slopes of straight lines defined by the pairs.
5. Determine K, the median of $\{k(p_i, p_j) | (p_i, p_j) \in PAIRS\}$
6. Let $SMALL := \{(p_i, p_j) \in PAIRS | k(p_i, p_j) < K\}$
   Let $EQUAL := \{(p_i, p_j) \in PAIRS | k(p_i, p_j) = K\}$
   Let $LARGE := \{(p_i, p_j) \in PAIRS | k(p_i, p_j) > K\}$
7. Find the set of points which lie on the supporting line h with slope K.
   Let MAX be the set of points $p_i \in S$, s.t. $y(p_i) - K \times x(p_i)$ is maximum.
   Let $p_k$ be the point in MAX with minimum x-coordinate.
   Let $p_m$ be the point in MAX with maximum x-coordinate.
8. Determine if h contains the bridge:
   If $x(p_k) \le$ and $x(p_m) > a$ then return ((k,m)).
9. H contains only points to the left of or on L:
   If $x(p_m) \le a$ then
   For all $(p_i, p_j) \in LARGE \cup EQUAL$, insert $p_j$ into CANDIDATES
   For all $(p_i, p_j) \in SMALL$ insert $p_i$ and $p_j$ into CANDIDATES
10. H contains points only to the right of L:
   If $x(p_k) > a$ then
   For all $(p_i, p_j) \in SMALL \cup EQUAL$ insert $p_i$ into CANDIDATES
   For all $(p_i, p_j) \in LARGE$ insert $p_i$ and $p_j$ into CANDIDATES
11. Return (BRIDGE(CANDIDATES, P)).
```

3. PDI

3.1 Algorithm for Overall PDI

```
1. Initialize the following:
      a. MCP, the set of ordered points representing the minimum convex polygon for the entire population range.
      b. A, the area of the minimum convex polygon defined by the hull.
      c. $S = [s_1, s_2, \ldots, s_{N_s}]$, a set of distances to calculate the bivariate k-function.
2. For l = 1:N
      a. $P^1 = \{all\ location\ measurements\ for\ animal\ l\}$
      b. $P^2 = \{all\ location\ measurements\ for\ all\ animals\ except\ animal\ l\}$
```

```
    c. $\left[k_1, k_2, \dots, k_{N_s}\right]_l = bivariatek(P^1, P^2, A, MCP, S)$
    d. Find mean and variance for each k1,k2, … over l.
```

3.2 Algorithm for Bivariate K-function

```
$\left[k_1, k_2, \dots, k_{N_s}\right]_l = bivariatek(P^1, P^2, A, MCP, S)$
  1. Initialize $K_{12}^1 = \left[K_{12,1}^1, K_{12,2}^1, \dots K_{12,N_s}^1\right] = 0$
                $K_{12}^2 = \left[K_{12,1}^2, K_{12,2}^2, \dots K_{12,N_s}^2\right] = 0$
  2. For i = 1:length of $P^1$
      a. For j = 1:length of $P^2$
          i. Compute distance between $(p_i^1, p_j^2)$
          ii. If distance is less than any value of s
              1. Compute $w(p_i^1, p_j^2)$ and $w(p_j^2, p_i^1)$, where $w(p_i^1, p_j^2)$ is defined as
                 the fraction of the circumference of a circle centered at $p_i^1$ and
                 crossing $p_j^2$ that lies within the minimum convex polygon
                  a. For each value of s greater than the distance
                     $K_{12,s}^1 \to K_{12,s}^1 + \frac{1}{w(p_i^1, p_j^2)}$
                     $K_{12,s}^2 \to K_{12,s}^2 + \frac{1}{w(p_j^2, p_i^1)}$
          iii. Else, distance is greater than all s values, do nothing
  3. $K_{12}^1 \to \frac{A}{m_1 m_2} K_{12}^1$ and $K_{12}^2 \to \frac{A}{m_1 m_2} K_{12}^2$
```

4. MCI

The global parameters are found by following the process described by Calabrese [1]. The log-likelihood function is given by

$$\ell = -MTrlogC_l - \frac{1}{2}\left\{\sum_{i=1}^{M} [\Delta x(t) - \boldsymbol{\mu_x}]^T C_l^{-1} [\Delta x(t) - \boldsymbol{\mu_x}] + \left[\Delta y(t) - \boldsymbol{\mu_y}\right]^T C_l^{-1} \left[\Delta y(t) - \boldsymbol{\mu_y}\right]\right\}$$

Where C_l (N by N), μ_x (N by 1) and μ_y (N by 1) are defined as follows

$$C_l := \begin{bmatrix} \sigma & \rho & \rho & \cdots \\ \rho & \sigma & \rho & \cdots \\ \rho & \rho & \sigma & \ddots \\ \vdots & \vdots & \ddots & \ddots \end{bmatrix}, \boldsymbol{\mu_x} = \mu_x \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}, \boldsymbol{\mu_y} = \mu_y \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$$

To find the parameters, the log-likelihood function is maximized. Because the C_l matrix is circulant, the C_l matrix can be re-expressed in terms of a uniform eigenvector and eigenvalues and the likelihood function maximized with respect to the eigenvalues. There is a closed form solution to the eigenvalues that maximize the likelihood function and they are as follows

$$\lambda_{0,max} = \frac{1}{2}\langle(\phi_0^T[\Delta x(t) - \boldsymbol{\mu_x}])^2 + (\phi_0^T[\Delta y(t) - \boldsymbol{\mu_y}])^2\rangle_t$$

$$\lambda_{+,max} = \frac{1}{2(N-1)}\langle\Delta x(t)^T\left(1 - \phi_0\phi_0^\dagger\right)\Delta x(t) + \Delta y(t)^T\left(1 - \phi_0\phi_0^\dagger\right)\Delta y(t)\rangle_t$$

where $\phi_0 = \frac{1}{\sqrt{N}}[1,1,\cdots]^T$ and $\phi_0^\dagger = \frac{1}{\sqrt{N}}[1,1,\cdots]$. Then, the values of the parameters can be obtained with the following relations (where the MCI is equivalent to ρ).

$$\mu_x = \phi_0\phi_0^\dagger\langle\Delta x(t)\rangle_t$$

$$\sigma = \frac{\lambda_0 + (N-1)\lambda_+}{N}$$

$$\rho = \frac{\lambda_0 - \lambda_+}{N}$$